`

CONTROL OF DATA CENTER COOLING USING DATA DRIVEN HEURISTIC APPROACH

A Dissertation
Presented to
The Academic Faculty

by

Mitchell Baxendale

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
The George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
August 2018

`

CONTROL OF DATA CENTER COOLING USING DATA DRIVEN HEURISTIC APPROACH

Approved by:

Dr. Yogendra Joshi, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Minami Yoda
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Peter Loutzenhiser
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: [May 13, 2019]

`

ACKNOWLEDGEMENTS

I would like to sincerely thank my advisor Dr. Yogendra Joshi for providing me the opportunity to obtain a graduate-level education to compose this thesis and serving as a guide to navigate decisions during my time and after graduation. I would also like to thank Dr. Minami Yoda and Dr. Peter Loutzenhiser for serving on my reading committee and providing valuable feedback.

I would like to thank Jayati Athavale for providing me mentorship in my research in the Data Center Laboratory, who served as my guide and inspiration for my work. I would also like to thank Kenneth Alexander and McKenney's for their assistance in the operation and maintenance of the Building Management System. I would like to thank Scott Robertson for guidance and assistance with the PI System. I thank Mike Mihuc for the internship opportunity at OSIsoft to more effectively understand how to use the PI System for my research, as well as serving as a great manager. I thank Vinayak Ruia and Dhaval Patel for assisting me with my duties in the Data Center Laboratory, and I thank the fellow members of Dr. Joshi's research group for providing feedback on my presentations as I prepared to present and defend my thesis.

I would like to thank my brothers at the Alpha Nu Chapter of Phi Kappa Sigma Fraternity, who remind me every day to have more fun. I would also like to thank my friends, who continue to provide me with much needed moral support. Finally, I would like to thank my mother, father, and sister for their unending love and support throughout all my life, and giving me the opportunity to attend the Georgia Institute of Technology.

`

TABLE OF CONTENTS

`

`

LIST OF TABLES

`

LIST OF FIGURES

`

LIST OF SYMBOLS AND ABBREVIATIONS

AF	Asset Framework
b	POD coefficient
BMS	Building management system
CFD	Computational fluid dynamics
COP	Coefficient of performance
CRAC	Computer room air conditioning
Damper%	Economizer damper percentage
DCL	Data Center Laboratory
DP	Dew point
DSC	Dynamic Smart Cooling
GUI	Graphical user interface
HT	Heat transfer
HVAC	Heating, ventilation, and air conditioning
ICU	Interface Configuration Utility
$\dot{m}$	Mass flow rate (kg/s)
ML	Machine learning
P	Power
PI	Process Information
PID	Proportional integral derivate
PDU	Power distribution unit
$\dot{Q}$	Heat removed from CRAC
POD	Proper orthogonal decomposition

`

ψ	POD mode
RH	Relative humidity
SDK	Software development kit
SHI	Supply heat index
T	Temperature (°C)
TCI	Thermal correlation index
VFD	Variable frequency drive
VSD	Variable speed drive

List of Subscripts

0	Reference
chiller	Chiller
CRAC	Computer room air conditioner
i,j	i-th or j-th component
in	Imported value
MA	Mixed air
OA	Outside air
out	Exported value
Rack	Server rack inlet
RA	Return air
SA	Supply air
Set	Set point
Target	Target value
WTS	Wireless temperature sensor

`

SUMMARY

Thermal management of data centers is becoming increasingly important in terms of reducing energy demand while ensuring server reliability. Much of the scope of the research surrounding thermal management of data centers revolves around temperature prediction and utilizing energy-saving technologies. The use of automatic control is widespread through many engineering applications, and as such many studies have utilized some level of control in the context of data centers. This thesis proposes a control method using the server inlet air temperature and cooling infrastructure data from the Data Center Laboratory collected using OSIsoft's PI System, importing the data into MATLAB to run conditionals and data analyses, and then exporting the data back into the PI System, where attributes configured with MODBUS two-way communication result in overwriting of the data in the Building Management System to achieve the desired control outcome. This controls approach is demonstrated in two case studies. The first demonstrates a heuristic model that enacts automatic control to maintain a target server inlet air temperature by changing return air temperature set points of a computer room air conditioning unit. The second case study controls the settings of the economizer in the computer room air conditioning unit to maintain a specified building envelope condition, while utilizing the economizer whenever appropriate to reduce the energy consumed by the cooling coil pump and supply fan. Additionally, real-time displays are updated to inform users of real-time conditions of the Data Center Laboratory. The primary outcome of this thesis is bridging the gap between temperature prediction models and automatic controls to result in dynamic, smart cooling of the data center.

`

Additionally, the control methodology applied in this thesis has wide-ranging applications to any components of the cooling infrastructure that are configured with two-way communication. Finally, the displays offer a more effective method of monitoring the real-time conditions of the data center. These displays should be applied to future control and/or temperature prediction methods to allow for more effective thermal management of the data center. In this way, automatic control is used with temperature predictions to effectively change the cooling infrastructure to user-defined settings, which may be updated to integrate more advanced temperature prediction models, software, building management systems, etc.

`

CHAPTER 1. INTRODUCTION

1.1 Background on Data Centers

Computer data centers are facilities that host large-scale IT hardware such as server racks for a given organization to store and process data. For example, Google, and Microsoft typically have their own data centers to handle the mass quantities of data that these companies must store and process. In addition, national laboratories or research institutions may house high power density data centers for supercomputers.

Data centers consume large amounts of energy relative to their size. Some of this energy of course powers the servers that are housed at the data center, but a considerable amount of energy is also dedicated to cooling the servers to keep them running at optimal conditions. Ultimately, nearly all the energy supplied to the interior of the data center is removed as waste heat. For instance, a single server rack may dissipate up to 37 kW as heat [1]. In addition, the computing capacity of data centers is increasing at an exponential rate. Top-performing supercomputers ran at ~1 Petaflops (1,015 floating point operations per second) in 2010, compared to ~50 Petaflops in 2015. This increase in computing capacity also entails an increase in the power dissipated by the data center, requiring massive energy demand to run and cool the servers. Data center heat densities increased from ~800-1,000 W/m^2 in 2002 to ~5,400-8,000 W/m^2 in 2008-2014. This power consumption becomes more dramatic for high-performance data centers: the average power consumption for each of the top 10 supercomputer data centers was ~8 MW in 2016 [1].

`

In a society placing increasing importance on energy efficiency, one must constantly consider design choices that would save energy. From an environmental standpoint, improving data center energy efficiency reduces demand for energy from power sources that emit considerable carbon dioxide to the environment. From an economic standpoint, the previous example of the power consumption of a supercomputer data center of ~8 MW translates to $7M in annual costs for continuous machine operation [1]. As such, thermal management of data centers is becoming an increasingly important topic for thermal science research and the data science industry.

1.2 Methods of Thermal Management of Data Centers

Typical "legacy" data centers are air-cooled using computer room air conditioning (CRAC) units. Cooling air is supplied via a raised floor plenum and returned via a ceiling plenum, as seen in Figure 1.1.

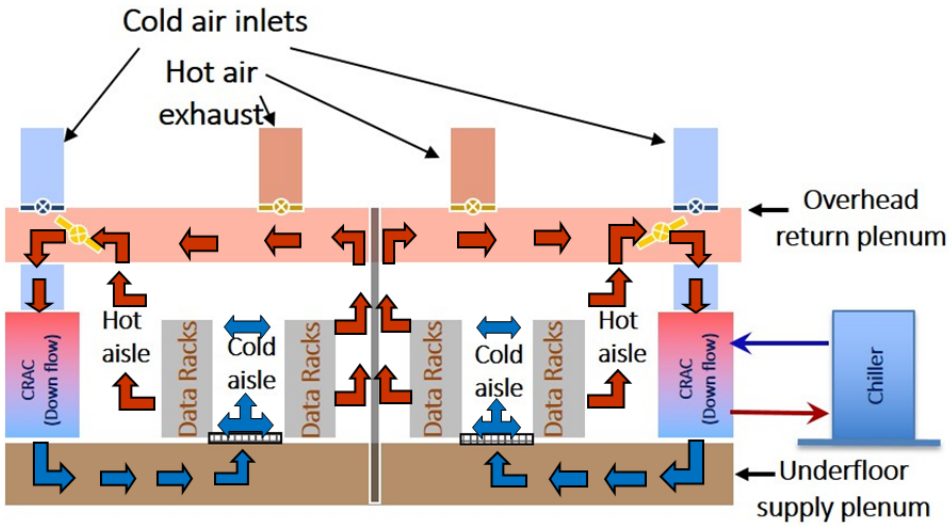

Figure 1.1 – Cooling air circulation in a typical air-cooled data center

`

High-performance data centers use a variety of methods to cool the servers, including chilled water on the server level, water-cooling on the chip level, or liquid immersion. While liquid cooling methods provide more effective cooling of the servers, the base cost of these methods is typically quite high, making them impractical for a typical smaller-scale data center. Furthermore, one must also consider the costs for converting legacy data centers to an updated cooling infrastructure, both in terms of facility modification costs and energy expenditure required to replace the existing system.

For thermal management of data centers, there are many design considerations over multiple length scales. The following scales are considered for a legacy air-cooled data center design. The smallest scale is the chip level in which heat spreaders, micro-heat exchangers, and conductive thermal interfaces are employed to more effectively dissipate the heat from the chip to the environment. On the server level, heat is dissipated via cold plates and/or heat rejection through the board surrounding the chip. On the chassis level, heat is dissipated using fans and larger heat exchangers. On the rack cabinet level, water-air heat exchangers are used. The room level utilizes flow of cold air to control the temperatures of the servers. Finally, the plenum level arranges the supply of the cold air to the room via chilled water pipes and perforated floor tiles. Figure 1.2 gives an idea of the scale of the different levels of data center thermal management.

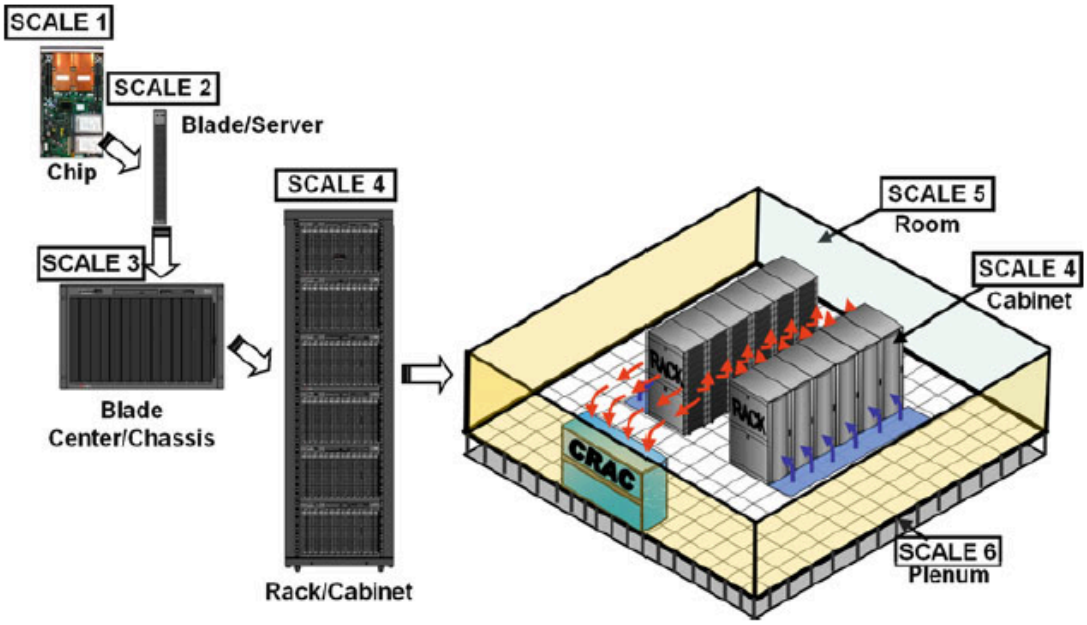

Figure 1.2 – Scale of different levels of thermal management in data centers [1]

1.3 ASHRAE Standards for Data Centers

Table 1.1 – ASHRAE Thermal Guidelines for Data Centers

Class	Equipment Environmental Specification for Air Cooling						
	Product Operations					Product Power Off	
	Dry-Bulb Temperature (°C)	Humidity Range, Non-Condensing	Maximum Dew Point (°C)	Maximum Elevation (m)	Maximum Temperature Change in an Hour (°C)	Dry-Bulb Temperature (°C)	Relative Humidity (%)
Recommended (Suitable for all 4 classes)							
A1 to A4	18 to 27	9°C DP to 15°C DP and 60% RH					
Allowable							
A1	15 to 32	12°C DP & 8% RH	17	3050	20	5 to 45	8 to 80

`

		to 17°C DP and 80% RH					
A2	10 to 35	12°C DP & 8% RH to 17°C DP and 80% RH	21	3050	20	5 to 45	8 to 80
A3	5 to 40	12°C DP & 8% RH to 17°C DP and 80% RH	24	3050	20	5 to 45	8 to 80
A4	5 to 45	12°C DP & 8% RH to 17°C DP and 80% RH	24	3050	20	5 to 45	8 to 80
B	5 to 35	8% to 28°C DP and 80% RH	28	3050	NA	5 to 45	8 to 80
C	5 to 40	8% to 28°C DP and 80% RH	28	3050	NA	5 to 45	8 to 80

ASHRAE maintains typical operating conditions of a data center, given in Table 1.1 [2]. The recommended range was created in the 2008 draft of Technical Committee 9.9, in order to "give guidance to data center operators on maintaining high reliability and also operating their data centers in the most energy efficient manner." The 2011 draft then introduced additional classes and operating conditions in light of different applications and energy-saving technologies. Data centers are denoted as operating within the A1-A4 class range, with A1 having the highest degree of control with regard to

`

environmental factors, while A4 the least level of environmental control. Figure 1.3 gives a psychrometric chart of these allowable ranges.

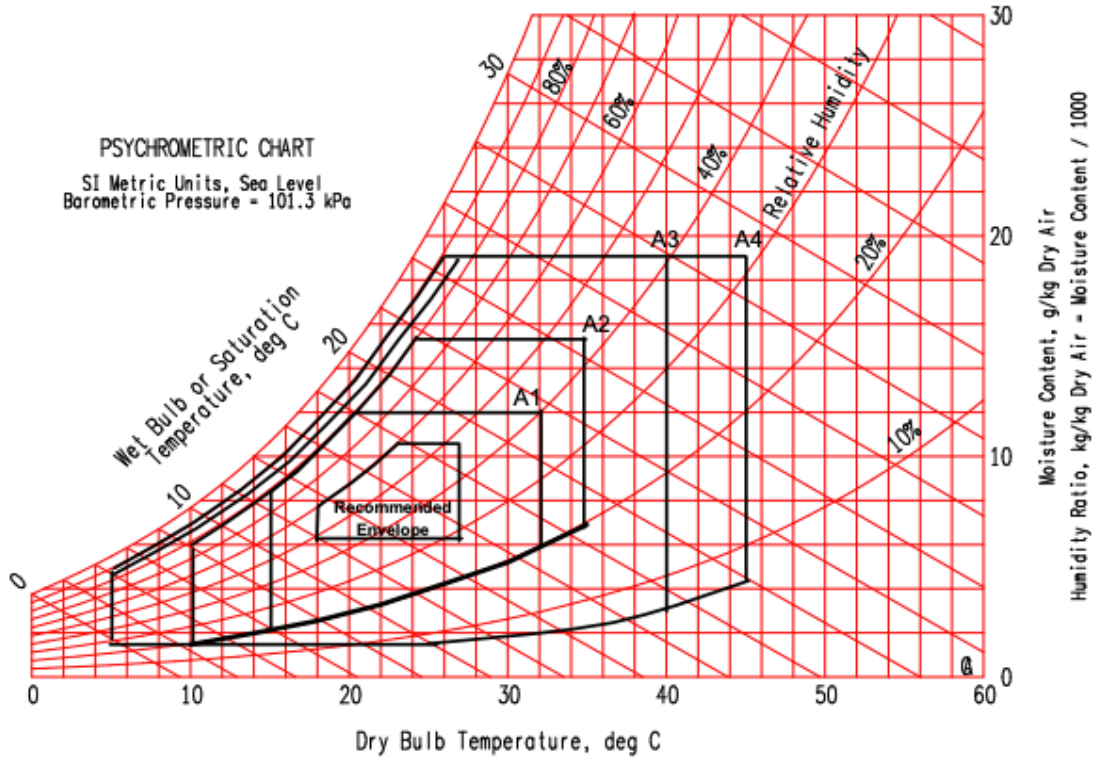

Figure 1.3 – Allowable temperature and humidity ranges for different data center classes [3]

1.4 Building Management Systems and Data Centers

Building Management Systems (BMS) are networks that control and monitor the equipment of a building, such as the heating, ventilation, air conditioning, lighting, etc. on an integrated system. BMS user interfaces may be simple LCD displays to interactive graphical user interfaces (GUI) that provide essential information to operators. BMS systems have the ability to control up to thousands of devices, making them highly effective tools for monitoring a building's performance and energy efficiency [4].

`

BMS technology is utilized in many mission-critical facilities, including data centers. For instance, in the context of data centers BMS systems can react to changes in cooling load, respond to events of equipment failure, and provide system operators with real-time conditions and alarms for the data center. Furthermore, use of BMS systems in data centers allow for more effective monitoring of the energy-intensive cooling equipment of the data center, reduce the risk of human error-caused outages, and facilitate effective alarms and notifications if outages occur [5].

In all, BMS systems are highly effective tools to monitor the real-time conditions of the equipment such as the cooling infrastructure and power supply of the data center, and are an effective way to remotely control the settings of the equipment.

1.5 Economization and Data Centers

Air and water-side economization techniques have been used in many building energy applications. However, with the expansion the ASHRAE guidelines for data centers in 2011, economization techniques are being adopted in data centers due to the increase in energy demand with the higher energy density of the modern data center. The primary advantage of utilizing economization, depending on time of year and geographic location, is the reduction or elimination of the chillers used to cool the data center [1].

Classes A3 and A4 of Table 1.1 were designed with the idea of using air-side economization, in which outside air is used to cool the servers as opposed to a traditional vapor compression cooling loop. As such, given the higher variability of the outside air conditions, the A3 and A4 classes have larger acceptability ranges of temperature and humidity [2]. When using the outside air, one must consider filtration, humidification,

`

and dehumidification techniques to ensure the reliability of the IT equipment. Implementing such techniques is costly, and may offset the benefits of cost savings from energy reductions [1].

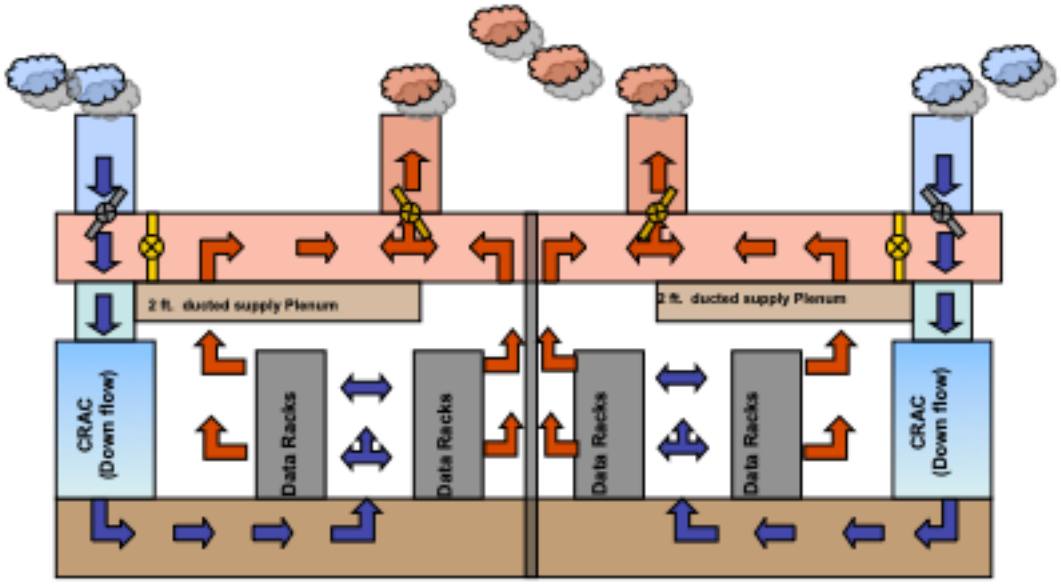

Figure 1.4 – Air-side economizer use in an air-cooled data center, with cooler outside air used instead of computer room return air

Figure 1.4 illustrates an air-side economizer in a traditional air-cooled data center, in which colder outside air is brought into the CRAC unit, thus reducing the energy demand of the cooling coil within the CRAC unit. Water-side economization is typically integrated with the cooled water loop of the chiller, where the water passes through additional coils in the data center room to cool the air [1].

Given the potential for energy savings with economization, economizer use is required by some building codes, including the Seattle and Massachusetts. Different studies have found energy savings of 40-45% using air and water-side economization

techniques in data center environments [1]. Based on these savings and wide adoption, the use of economizers in the data center environment is becoming more commonplace.

1.6 Thesis Preview

The goal of this thesis is to propose a control and prediction methodology for use in the data center to effectively address the concerns for more effective thermal management of the data center and ensuring energy efficiency of the data center instruments. This thesis focused on normal operation of the data center, and as such will not take into account events of catastrophic failure. Potential integration of such events is discussed in Chapter 7.

Chapter 2 of this thesis covers the relevant literature related to the control method presented in the thesis and the corresponding case studies. Chapter 3 presents the infrastructure of the Data Center Laboratory (DCL) in terms of the equipment and data collection methods currently employed. Chapter 4 presents the control methodology used to import and export data resulting in controls of the cooling infrastructure. Chapter 5 presents the methodology, results, and discussion of a case study controlling the temperature set point of the CRAC unit to result in reaching a target server inlet air temperature. Chapter 6 presents the methodology, results, and discussion of a case study controlling the economizer damper position of the CRAC unit to optimize economizer use while maintaining the supply air temperature of the CRAC unit. Chapter 7 offers closing remarks, including key contributions of the thesis and suggestions for future research.

`

CHAPTER 2. LITERATURE REVIEW

2.1 Temperature Prediction in Data Centers

Much of the surrounding research of thermal management of data centers is based around temperature prediction models. Predicting temperature distributions within data centers allows for more accurate models of server inlet air temperature over time, which may be used to provide more targeted activation of cooling infrastructure. Many approaches for predicting server inlet air temperature distributions are based on physics-based computational fluid dynamics/heat transfer (CFD/HT). Heuristic models are based on system identification simulations. Data-driven models include proper orthogonal decomposition (POD) and machine learning (ML) models.

Given the infrastructure of the Data Center Laboratory (DCL) with the data archiving capabilities of the PI System, discussed in Chapter 3, it is desirable to follow a temperature prediction model that is based on past and current data. The computation times of typical CFD/HT models and the complexity of ML models did not fit into well with the desired infrastructure of the PI System and MATLAB, further discussed in Chapter 3. As such, the focus of this thesis revolves primarily around heuristic and data-driven models, with emphasis on simple, fast, time series-based temperature prediction models to provide for more active controls, as opposed to robust temperature prediction.

2.1.1 Heuristic Models

Li et al [6] utilize an auto-regressive model that relies on real-time environmental data to predict inlet and exhaust temperatures based on relationships derived from

`

thermodynamic principles and past data trends for specific parameters. Using this method resulted in accurate temperature predictions for windows ranging from 5-20 minutes based on training data sets ranging from 15-90 minutes.

Chen et al [7] developed a model based on recorded data with a transient CFD simulation to calibrate boundary conditions through a least-squares method as a training data set for real-time temperature predictions. Using this method on a single rack and a five-rack production data center, the prediction model typically resulted in error less than 1°C, except for an error of 6°C during an AC failure. Chen also noted the increase in error as prediction horizons increased.

Both of these methods provide a basis for rapid computation based on real-time data. However, both of these models need some sort of experimental data or CFD prediction in order to properly estimate parameters. As such, these heuristic methods must be coupled with some sort of systems identification experiment of inputs and outputs in the data center environment.

2.1.2 Data-Driven Models

Much of the data-driven models that do not involve ML techniques revolve around POD, also known as Karhunen-Loeve decomposition, given in Equation 2.1 [8]

$$T = T_0 + \sum_{i=1}^{m} b_i \psi_i \tag{2.1}$$

where T is a temperature field, T_0 is a reference temperature, b_i is the calculated POD coefficients from Galerkin projection, interpolation, or flux matching [8], and ψ_i is the calculated POD modes from CFD/HT simulations or experiments. POD is used as a

`

means of calculating a temperature field through numerical behavior of collected data, as opposed to the physics-based models of CFD/HT or the system identification methods of heuristic temperature prediction [8].

Samadiani [9] used a POD model with parameters obtained from CFD/HT simulation to predict temperature fields for given heat loads and CRAC airflow speeds in a data center environment. This method resulted in an average error of 1.24°C, with a computation speed ~150 times faster than CFD/HT simulation [8]. Samadiani et al used the POD framework on a larger scale, instead calculating POD parameters through experimental results, and predicted temperature fields for different CRAC utilization loads. Here, the average error was 0.68°C, with a maximum local error of 8°C.

2.2 Controls in Data Centers

Automatic controls and feedback loops are utilized extensively in all facets of mechanical design in order to mitigate error. In the context of data centers, this control has been applied to the different levels of thermal management, from server-level control up to plenum-level control. This thesis is primarily concerned with room level control with changing settings of the CRAC units given the existing infrastructure of the BMS. However, multi-level studies that incorporate both room-level and localized control were investigated.

2.2.1 Fan Control

Joshi and Kumar [1] analyzed the power consumption of the data center using a feedback loop in conjunction with variable frequency drives (VFD) of the CRAC units to

`

control the cooling rate, rather than setting the VFDs at a fixed speed of 100%. By varying the speed from 60% to 100% over a day of operation, they realized 47 kWh in savings, or 6% of the total input power to the CRAC and chiller.

Ahuja et al [10] sought to reduce the mismatch between the required airflow for the IT equipment and the airflow supplied from the facility fans to prevent air recirculation. This was done by controlling the fan speeds of the individual CRAC units, resulting in a reduction of energy consumed by the CRAC units by 77%, and chiller equipment, as well as a lower Power Usage Effectiveness.

Boucher et al [11] compared control methods based around the supply heat index (SHI), a ratio of temperatures at the server inlet/outlet minus the temperature of the adjacent rack. Varying the CRAC fan speed while server heat load and vent position were kept constant resulted in a decrease in SHI up to around 60%. This means that increased CRAC fan speed results in lower temperatures as the effects of recirculation are reduced, however past 60% has little effects and thus wastes energy from blowing. Due to the nonlinear effects, control should be coupled with more complex controls such as CFD, neural networks.

2.2.2 Temperature Set Point Control

Bash et al [12] propose a distributed sensor network that interacts with the attributes of CRAC units in order to implement dynamic thermal management, using a Thermal Correlation Index (TCI) used to quantify the change in temperature readings at the rack level in response to a temperature range at the CRAC level, given in Equation 2.2.

`

$$TCI_{i,j} = \frac{\Delta T_{WTS,i}}{\Delta T_{CRAC,j}} \quad (2.2)$$

These data were applied to a Dynamic Smart Cooling (DSC) controller, which takes the TCI and temperature sensor data with a rule-based filter, proportional-integral-derivative (PID) compensator, and a flow scheduler to determine the optimal temperature set point and flow rate. By using this method as opposed to a traditional compensator control method, power savings from experiments ranged from 40-50%.

Zhang et al [13] controlled CRAC supply air temperature based on server inlet temperatures as opposed to modular temperature sensors around the lab environment, resulting in an increase of the supply air temperature to reduce the PUE. In addition, server-level temperature monitoring combined with CFD modeling allows for more effective thermal management.

Boucher et al [11] ran multiple experiments focusing on the results of controlling different aspects of data center cooling equipment. When controlling the CRAC supply air temperature, keeping fan speeds and vent positions constant, results in a relatively linear relationship between the CRAC supply air temperature and the air inlet temperature, illustrated in Figure 2.1.

`

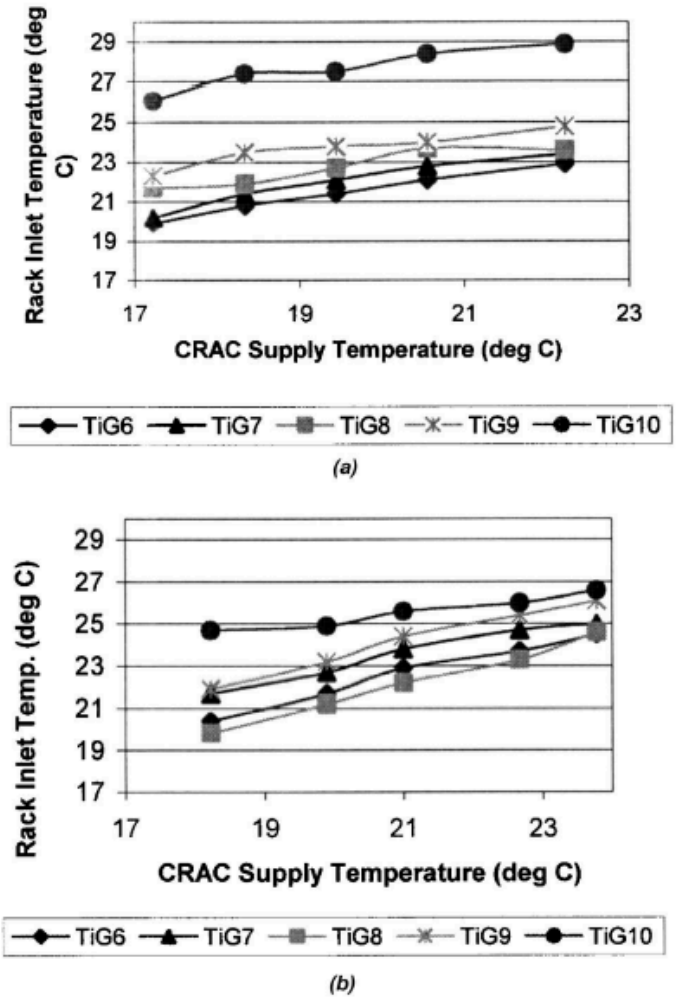

Figure 2.1 – Rack inlet temperatures for (a) 33% VFD and (b) 66% VFD for Boucher's experiments varying the supply air temperature on the effects of rack inlet air temperature [11]

2.2.3 Multi-Level Control

Patel et al [14] propose a "Smart Data Center" which localizes cooling and workload allocations and makes adjustments via component control. They also note the differences in using a centralized feedback control vs. delocalized agents resulting in multi-level control methods. Furthermore, a basic GUI of monitoring the conditions of the data center is also presented.

`

Wang et al [15] make use of local controls using a layered control scheme in which tile openings and CRAC blower speeds are controlled to provide cooling on local and zonal levels. By implementing zonal control, power consumption was reduced.

While the current infrastructure of the building management system (BMS) at the DCL is not suited towards localized cooling control, perhaps such components could be configured with the PI System and MODBUS in a similar fashion to the BMS to allow for the reading and writing of data on a localized level.

2.3 Approach

Given the capability of the BMS to overwrite the fan speeds and temperature set points of the CRAC units in the DCL (discussed in Chapter 3), these methods of fan and temperature set point control could be applied to the DCL. While the current infrastructure of the BMS is not suited towards localized cooling control, perhaps such components could be configured in a similar fashion to the BMS to allow for the reading and writing of data on a localized level. Given the linear relationship between controlling the temperature set point of the CRAC and the rack inlet air temperature [11], one could liken the slope of this line as the TCI [12]. Given this linear relationship, it is therefore desirable to use an auto-regressive prediction model similar to Li's [6] in which linear temperature predictions are used to calculate optimal temperature set points for the CRAC unit.

`

CHAPTER 3. THE DATA CENTER LABORATORY AT GEORGIA TECH

The Data Center Laboratory (DCL) at the Georgia Institute of Technology is one of the few university laboratories in the United States dedicated to analyzing the thermal distributions, HVAC design, and energy consumption of data centers. The data center is a typical legacy design with 1 m raised floor plenum with perforated tiles and return ducts in the ceiling. The lab also features six CRAC units (two upflow and four downflow) and three power distribution units (PDUs). The DCL is divided into two 56 m^2 zones: experimental zone with space up to 14 server racks and a computational zone with 16 server racks. The DCL is rated to a ~4470 W/m^2 power density for the lab as a whole. In order to monitor and store large quantities of data from temperature sensors, CRAC units, etc. one must employ some method to read and store the data, keeping track of the multitude of sources of data. The DCL relies on the PI System software from OSIsoft for this data gathering and storage. Figure 3.1 gives a plan view of the DCL.

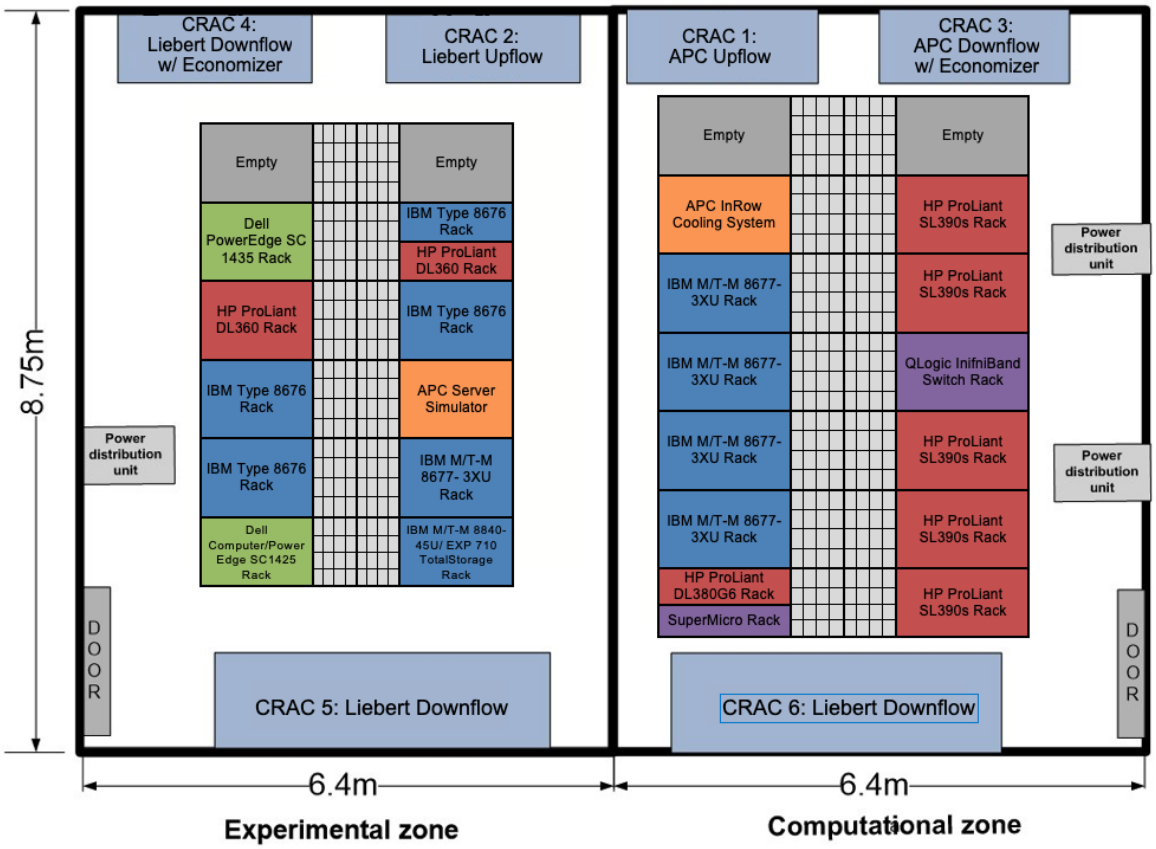

Figure 3.1 – Plan view of the Georgia Tech Data Center Laboratory (DCL) server racks, CRAC units, and PDUs

3.1 Infrastructure

The air temperatures at the server inlets are measured by muRata SN802GRC-4DM-X MODBUS wireless temperature sensor (WTS) modules. Each module has four temperature ports, J1-J4. Figure 3.2 the typical orientation of a server rack with the accompanying temperature sensor location.

`

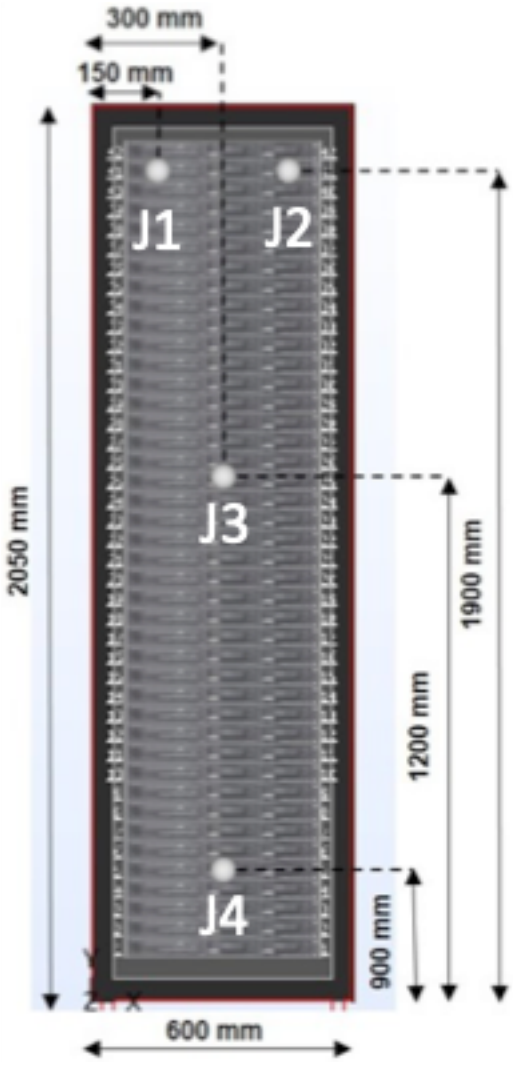

Figure 3.2 – Server schematic with temperature sensor port positions: J1 upper left, J2 upper right, J3 middle, J4 lower

3.2 The Building Management System

The Building Management System (BMS) is a Java-based program from McKenney's Inc. that is hosted on a local computer in the DCL that collects data associated with the CRAC units and power distribution units. The data for the CRAC units are displayed in a GUI that gives a layout of the CRAC unit of interest with corresponding information such as supply air temperature, return air temperature, etc.

`

arranged as illustrated in Figure 3.3. Other data are displayed in tables, and each data point shown as a square icon that gives a time series graph of the attribute of interest. Each CRAC unit display is unique, with the exception of CRAC units 5 and 6, which are the same unit model.

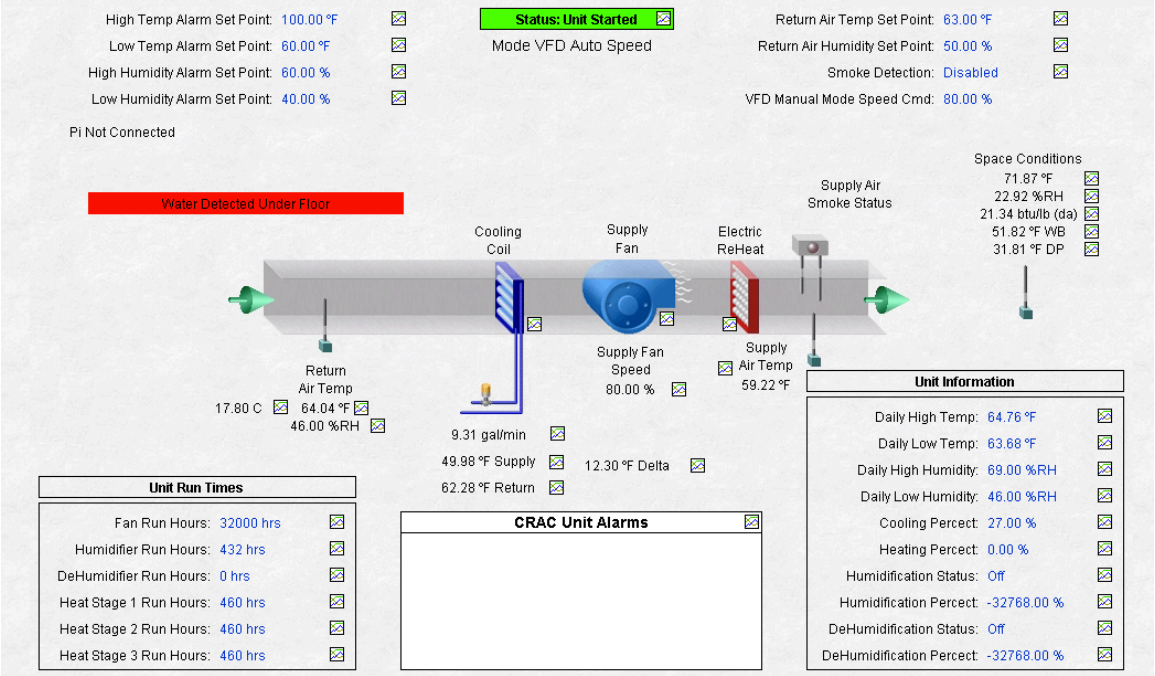

Figure 3.3 – BMS display for CRAC 5, with schematic for airflow, set points, and accompanying information

Using this program one can assign the set point values of the CRAC units of the DCL by right-clicking on the set point and writing a new value in a text box. This saves time in users manually changing the set points in the CRAC units themselves. Additionally, CRAC units 2 and 3 feature economizers that can only be controlled through the BMS. Using the BMS, users may define the settings of the economizer, such as the damper position and the cut out set point. Figure 3.4 gives a display of CRAC 2.

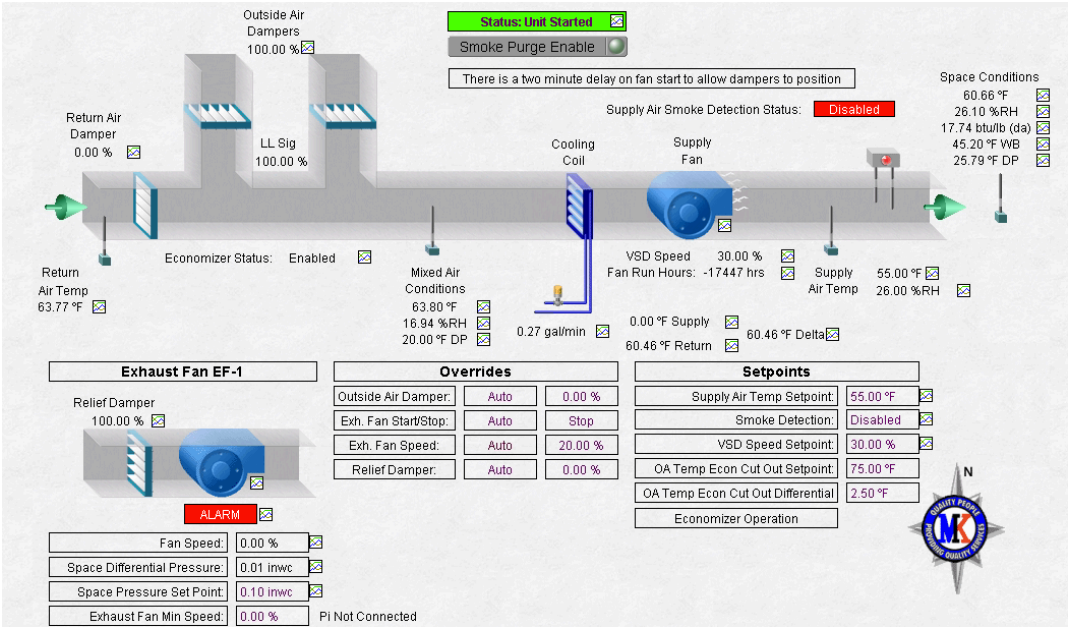

Figure 3.4 – BMS display for CRAC 2 with schematic for airflow, economizer operation, set points, and accompanying information

Additionally, the BMS has power consumption information from the PDUs and CRAC circuits, illustrated in Figure 3.5. In this way, the BMS is a highly effective tool in monitoring and controlling the cooling and power data of the DCL.

`

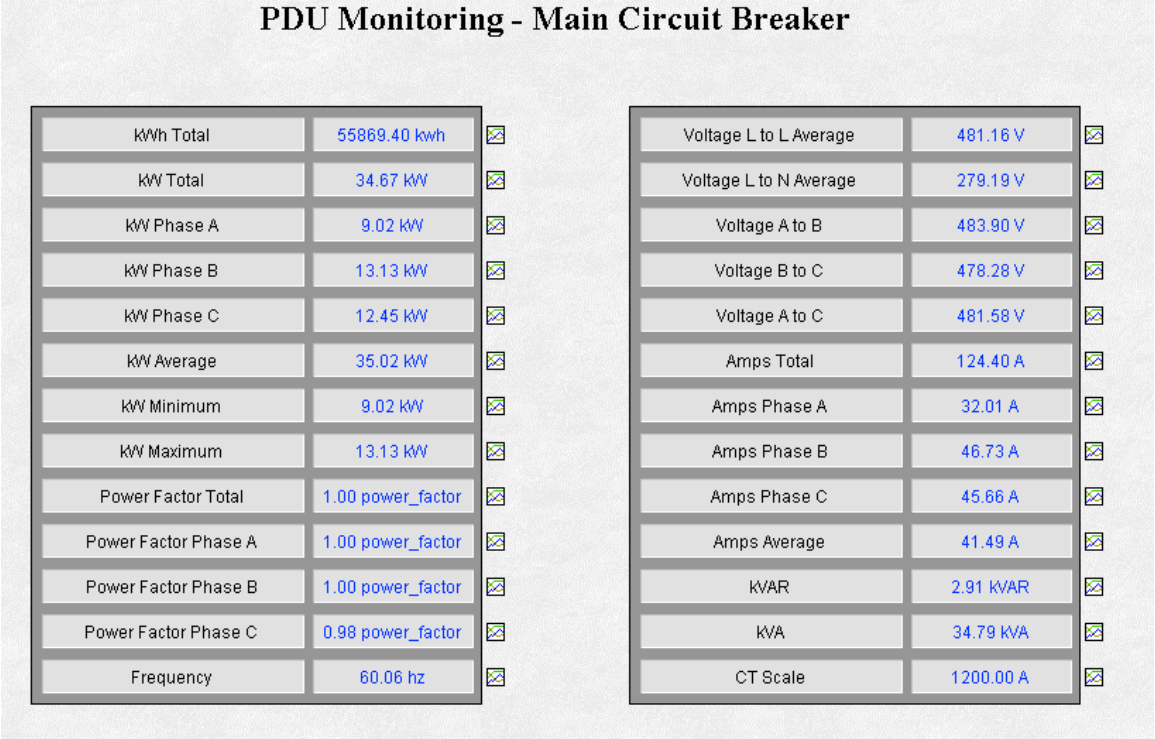

Figure 3.5 – BMS display for PDU monitoring

3.3 PI System

The PI (Process Information) System from OSIsoft is a series of programs that allow for the gathering archiving, and access of large amounts of data over many sources and devices. The PI System supports over 450 different data interface types, and features many tools for organizing, analyzing, and storing large amounts of historical data. The main components of the PI System are the PI Interface, the PI Data Archive Server, and the PI Asset Framework Server. The PI Interface takes data from multiple sources of data, and then synthesizes the different sources of data into a common "language", where it is then stored in the PI Data Archive Server. From the PI Data Archive Server, the data is translated into a user-accessible database, the PI Asset Framework Server. Once the data is stored in the Data Archive Server or an AF Database, it may be accessed from

`

many different clients to visualize and import it. These clients may be programs from OSIsoft, or an add-on to an existing program. Figure 3.6 shows a flow diagram of how data is processed and displayed in the PI System. [16]

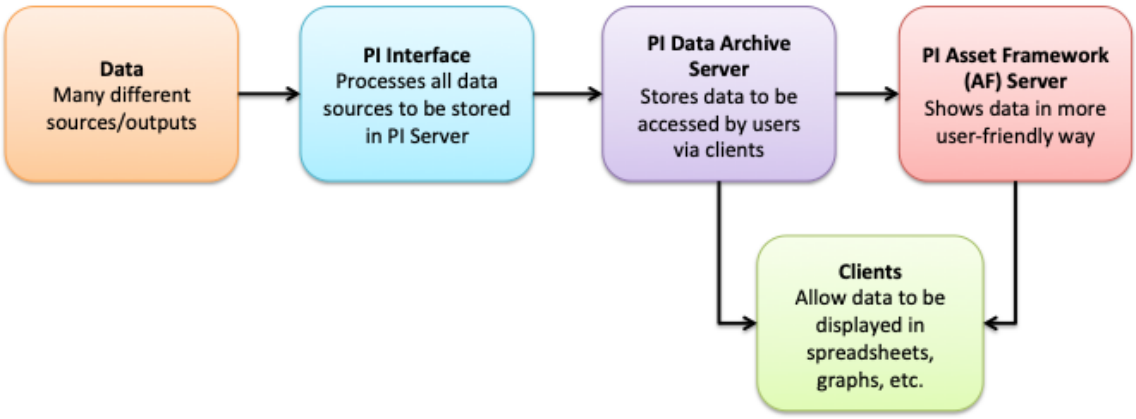

Figure 3.6 – Flow diagram of data from instruments into the PI System software

The PI System is used widely in many engineering applications as a means of storing and organizing large quantities of data. In the context of data centers, companies such as eBay and PayPal standardized their data centers using the PI System, Dell uses the PI System in their modular data centers, and Maya and Energy Metrics use the PI System as part of their data organization solutions for data centers (S. Robertson, personal communication, May 2, 2019).

3.3.1 PI Asset Framework

PI Asset Framework (AF) server is an accessible database that displays specific components with accompanying information. PI elements are the components and sub-components that make up organizational hierarchy of the PI AF database. PI attributes are pieces of information that describe the elements. PI points (or PI tags) are the specific PI

`

attributes that store data within the PI System. Other PI attributes for an asset include metadata on a specific attribute. PI attributes also may have sub-attributes, such as the name and description of the tag. This naming convention of elements, attributes, and points will be maintained in this thesis [17]. For instance, within a data center a PI element could be a specific CRAC unit, while a PI attribute would be the return air temperature of that CRAC unit. Specific elements may also have templates with pre-assigned attributes, which is useful for collecting data from identical units such as multiple wireless temperature sensor modules.

The PI AF server also allows for the storage of "future data", which is data that is stored at future time stamps. Future data is generated for a specified time range, and stored in a "future archive" database, separate from the PI Data Archive Server such that predicted values from previous times are saved and don't overwrite or are overwritten by measured values. Future data may be overwritten by other data designated with the same future data PI point and timestamps, for example a forecast for an attribute updating with new values at a given future time. Meanwhile, forecast data that is not overwritten is still saved in the future data archive.

PI AF also has an analyses tab that allow for equations, conditionals, data and time retrieval, etc. internally within PI AF. Analyses are useful for relatively easier calculations within PI AF that can be calculated in real time. Additionally, the outputs of the expressions of a certain analysis may be mapped to specific attributes of the element, allowing for easy calculation without the need for outside software. However, analyses are limited in their scope. For instance, analyses cannot create forecasts of future data,

`

and the structure for conditionals is fairly simple. Nevertheless analyses are a powerful tool for performing calculations of PI AF data in real time.

3.3.2 *PI Vision*

PI Vision is an html-based program that allows for visualization of data from the PI AF Server in tools such as tables, figures, and graphs in individual displays. From a web browser, one simply has to connect to the specific server that hosts the PI System, select a database within the system, and then select or create displays based on the attributes of the database's assets. Displays may be linked, for example from a summary display to a detail display. Furthermore, displays may be saved as collections, analogous to templates in the AF server. A collection features the same outputs of data and may be applied to assets of the same type. For example, one could create a collection for a specific CRAC unit, and that collection may be applied to all of the other CRAC units to obtain the same type of data from each CRAC unit. PI Vision may display real-time data, historical data, or future data by adjusting the time scales.

Another benefit of PI Vision is the ability to apply condition-based monitoring. One may apply color-coded states for data that correspond to the operating condition of a specific attribute. For example, one could configure a multi-state that defines an acceptable range of temperatures for a temperature sensor, however once the limit is exceeded the value corresponding to the exceeded temperature will change color, indicating to a user that an action must be done to address the issue. Unfortunately, this condition-based monitoring is only applicable to numerical data. Regardless, this condition-based monitoring allows for more dynamic, intuitive displays.

`

3.3.3 PI System and MATLAB

MATLAB from MathWorks is a piece of software commonly used for programming and data analysis. MATLAB is known its prevalence for engineering applications in research, academia, and industry, as well as its ability to perform large-scale data analysis with sophisticated, user-defined functions and scripts. MATLAB also has several add-ons that allow it to be integrated with other software. The versatility of MATLAB lends itself well to be integrated with the large amount of data stored in the PI System. In addition, the ability to import tabulated data from MATLAB back into the PI System would result in robust forecasts, displays, and notifications to allow for effective monitoring.

Real-time data may be obtained using the PI AF Software Development Kit (SDK), which is installed with the PI System. PI AF SDK allows for programmatic access to PI System data using a programming language of choice, including MATLAB. The data from PI AF is loaded as a .NET assembly. One may obtain templates, elements, and attributes from the assembly by calling variables and data within the assembly. Additionally, one can obtain the values and timestamps of a particular attribute for a given time range. Finally, one can write data back into the PI System as real-time data or future data, which is particularly effective for detailed notifications and forecasting. Drawbacks of using PI AF SDK is that one has to have MATLAB on the same computer as the PI System, and that running PI AF SDK on a remote desktop PI server could result in some timeout errors. Despite this, PI AF SDK is a simple, effective way of obtaining real-time data from the PI System into MATLAB. [18]

`

3.4 PI System and the Data Center Laboratory

Figure 3.7 and Figure 3.8 give the typical elements and attributes associated with the CRAC units and WTS units, respectively. The attributes for the CRAC units correspond to the data points read from the BMS, such as supply and return air temperatures, temperature set point, etc. The CRAC element templates feature toggling for the set points, which will be discussed in Chapter 4.

For the WTS units, there is the temperature value for each J1-J4 port, along with corresponding metadata on the location of the port. Furthermore, there is an attribute for the temperature forecast configured as future time, as well as strings for status messages of the temperature sensors. These data are read directly from the WTS modules, which feature unique IP addresses.

These points from the DCL are configured to send data to the PI System interface node using the PI Interface Configuration Utility (ICU). Different points of data are assigned different scan classes, or request speeds, defined by the user. For the BMS, since there are over 300 different points of data all being processed by the ICU, these scan classes are important for assigning priority to certain variables to be read first, for instance the read values of the set points. Meanwhile, this issue does not exist with the WTS modules, which have unique IP addresses that may be read instantly.

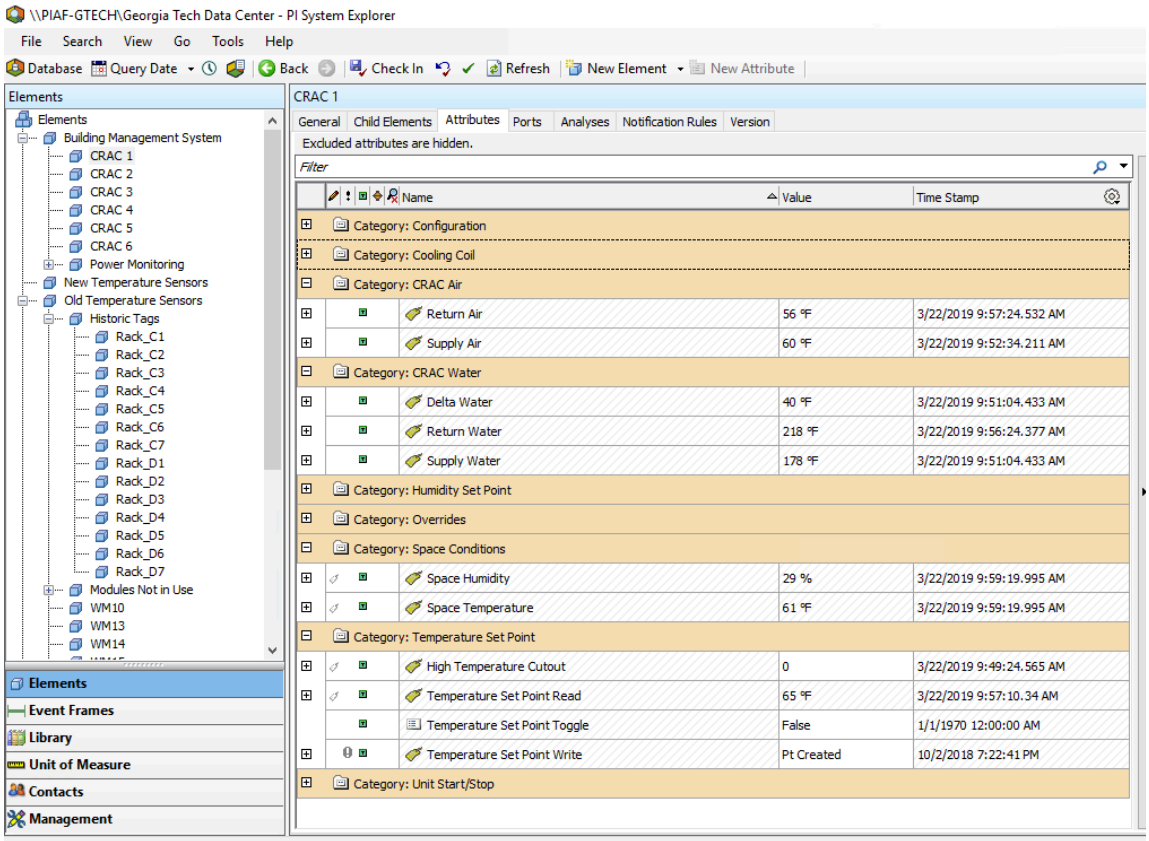

Figure 3.7 – PI element template for CRAC units in the DCL, with typical information provided from the BMS displays along with writing registers for set points

`

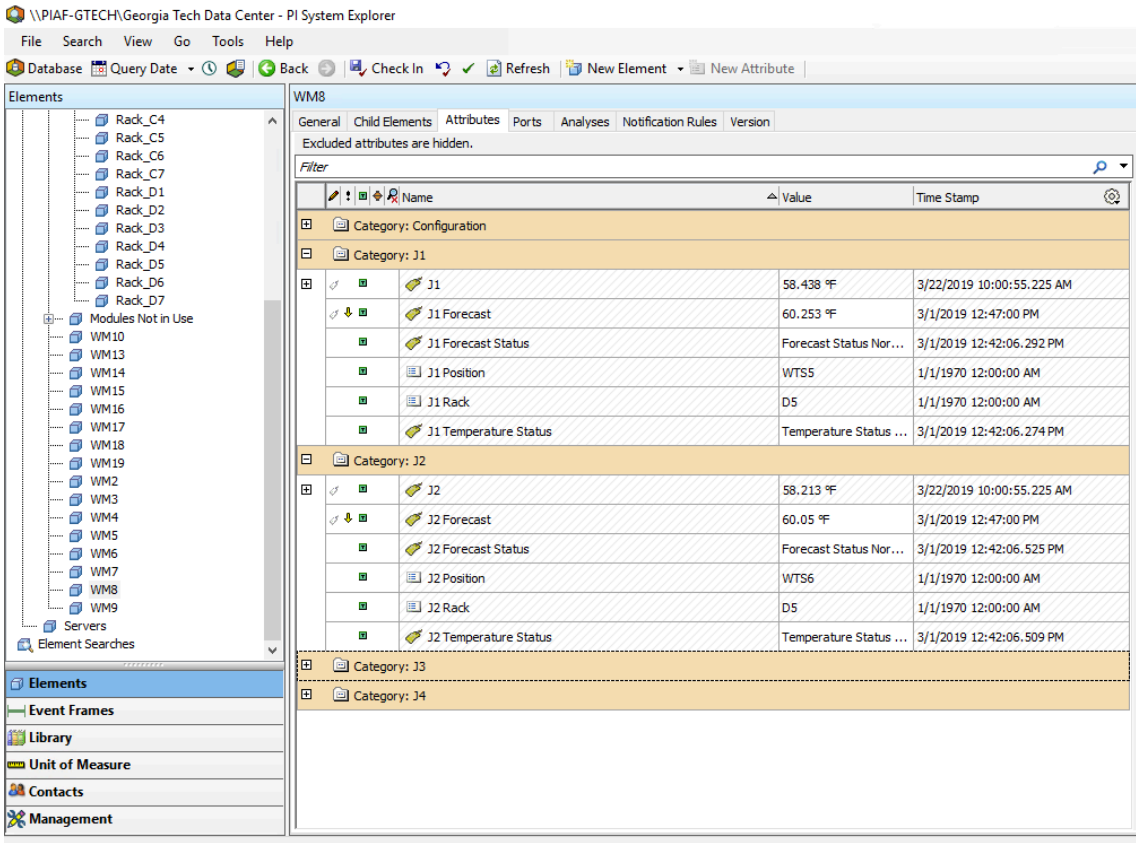

Figure 3.8 – PI element template for WTS units in the DCL, with temperature forecasts saved as future data accompanying error messages

3.5 Writing Data to the Building Management System

Data center control is typically done through manually changing the settings of the BMS interfaces or by manually changing the settings of the instruments themselves. However, there has been growing interest in dynamically changing the settings of the BMS without the use of manual operators. For instance, Google previously utilized a machine learning algorithm to communicate optimal set points to data center operators, however recent modifications allow this algorithm to make adjustments to the cooling infrastructure automatically [19], using MODBUS communication and Google Cloud Platform technologies (S. Robertson, personal communication, May 2, 2019). While this

`

method of control has not been widely applied to data centers, it could work with a BMS that has MODBUS communication for set points.

While there is no default two-way communication between the PI System and other pieces of equipment, one can configure PI points to have MODBUS source, which will allow for reading and writing of data between different registers. In this way, registers corresponding to set points may be overwritten in the BMS as long as those specific attributes are configured with MODBUS. The set points for the BMS in the DCL were configured to allow for this MODBUS two-way communication, and as such this thesis utilizes this as the primary means of control of the set points of the CRAC infrastructure.

For the PI System to access the BMS, bits 0 and 1 of a control register are continuously toggled by writing values of 1 (bit 0) and 2 (bit 1) to the control register attribute in PI AF. The control register ensures that the PI System is actively connected to the BMS. When the PI System is not connected to the BMS, the set points will revert back to their previous values before PI connected to the BMS. From there, the user may select different set points of the BMS to control, including temperature set point, exhaust fan speed, economizer settings etc. Each of these set points has an associated 16-bit integer that is added to the control register. For instance, if one wanted control of the temperature set point, one would add the associated integer value of 8 to the control register attribute, resulting in a continuously alternating value of 9/10 when adding to the toggling of bits 0/1. Adding other attributes to toggle to the control register results in the ability to control multiple attributes at once. Finally, once PI has control of the BMS the attribute of interest may be overwritten using the writing register attribute of the specific

`

set point of interest. In this way, users may control the BMS without having to physically access the BMS machine.

3.6 Desired Changes to Data Center Laboratory

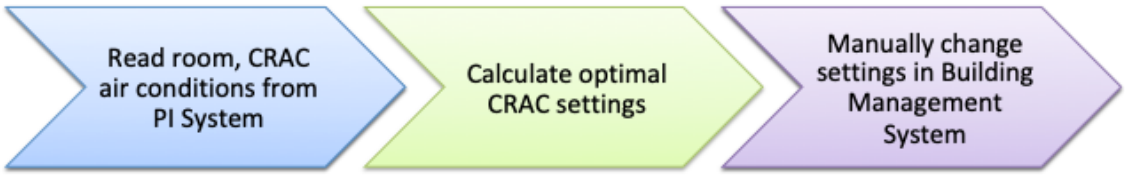

Figure 3.9 – Process flow of changing CRAC settings, before control methodology

Figure 3.9 gives a flow diagram on how problematic temperatures are addressed in the DCL without using the proposed control methodology. Prior to this thesis, the PI System was only used to store and display past and current data. Temperature data from the PI System was read by the user and then equipment such as the CRAC units were adjusted manually in the BMS in order to address problematic temperatures or energy consumption. Therefore, it was desirable to find some intermediary program to read the data from the PI System in real time and use internal trending and conditioning to adjust the CRAC settings automatically. Figure 3.10 gives the existing interface of data between the instruments of the DCL and the PI System.

`

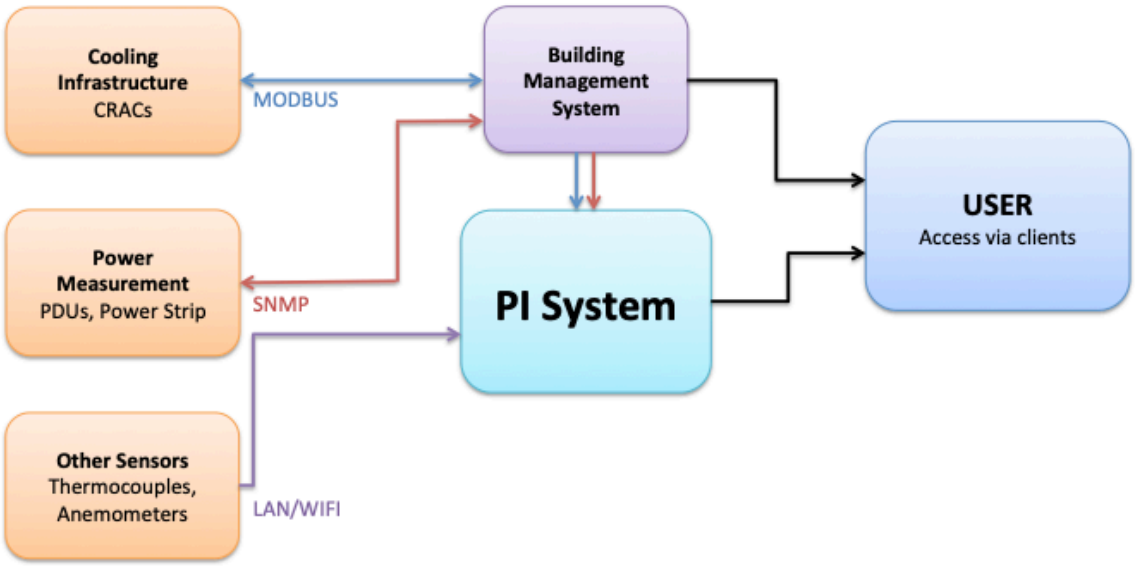

Figure 3.10 – Data flow between DCL instruments and PI System, before control methodology

Using PI AF SDK with MATLAB to import/export data to the PI System, internal trending and conditionals coded in MATLAB would result in writing data such as optimal cooling set points to the components with two-way communication. Additionally, such trending could generate forecasts and error messages to update PI Vision displays, resulting in effective, efficient monitoring of conditions within the DCL. Figure 3.11 shows the modified data flow to the PI System using this MATLAB-based code.

`

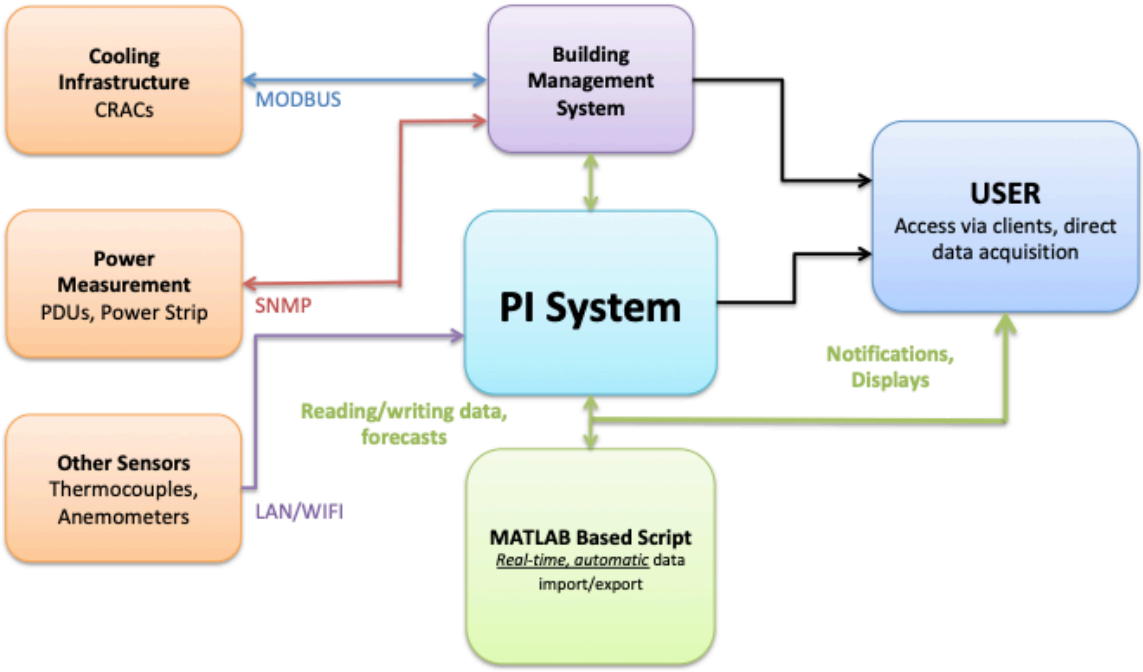

Figure 3.11 – Desired data flow between DCL instruments and PI System, using new control methodology

For equipment that can be controlled remotely, such as the BMS set points configured via MODBUS, this program would enact automatic controls once a certain acceptability criteria is violated. For components that do not have communication capabilities, this program would trend the data internally to create highly detailed PI Vision displays so that the user would not have to physically go into the PI System to read the data and determine the optimal settings to correct the issue. Figure 3.12 gives a process flow of the data moving between the different components of this controller.

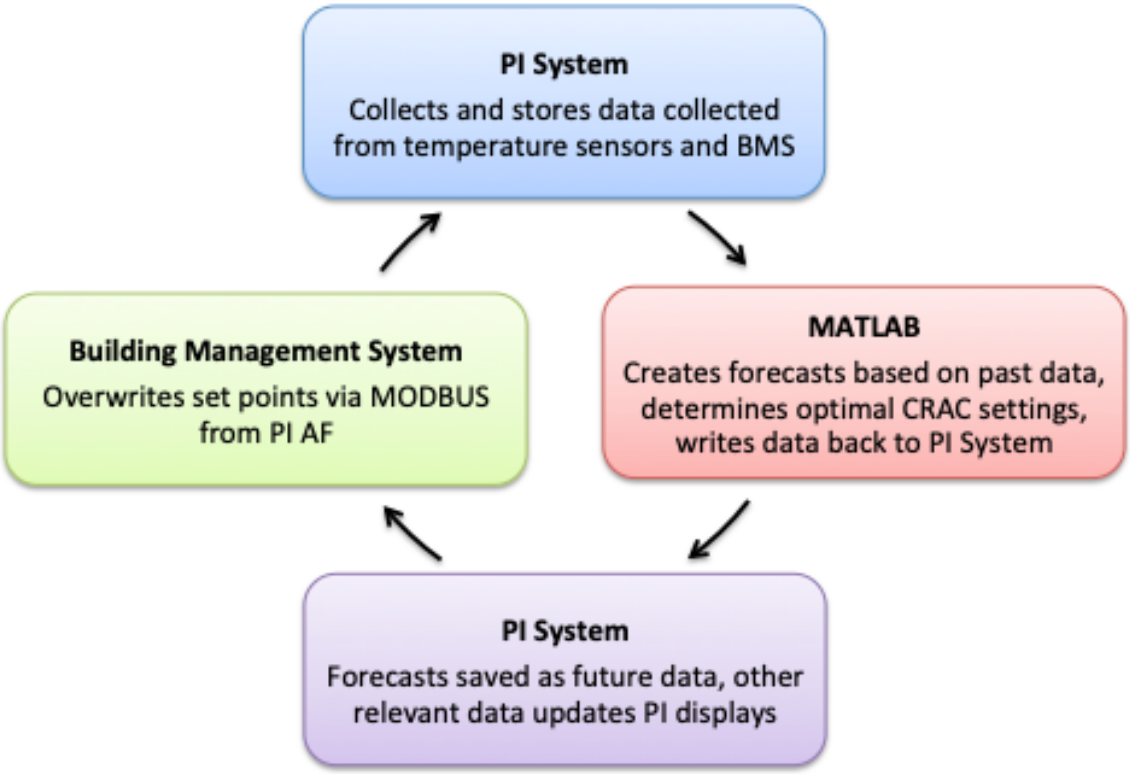

Figure 3.12 – Process flow of data cycle of proposed MATLAB-based controller

`

CHAPTER 4. CONTROL METHODOLOGY AND MONITORING

4.1 PI AF Analyses

The process of toggling specific set points within the BMS by writing values to the control register was streamlined using an analysis within PI AF, shown in Figure 4.1. A list of Boolean true/false attributes were created to correspond to the set points and specify whether the user wants to take control of those specific set points. The analysis tests to see if these attributes are true/false, and if they are true the analysis will add the associated 16-bit value of each set point to the control register. Furthermore, the analysis continuously writes values of 1 and 2 to the control register in fixed intervals to automate the connection to the BMS.

Figure 4.1 – PI AF Analysis in which values are added to the toggle register depending on if certain attributes are "True" or "False", resulting in continuous toggling of specific attributes in the BMS using PI System

`

4.2 MATLAB and Importing/Exporting PI System Data

Two functions were created in MATLAB in order to get numerical PI System data in and out of MATLAB. The "import" function takes the specific element and attribute of interest, along with the data type and a specified time range, and reads the data from the PI AF SDK .NET assembly and sorts the data into a vector of values and timestamps (in MATLAB serial time notation) that correspond to each value. The "writer" function works in reverse: the vectorized data is written back into the .NET assembly back into PI AF using PI AF SDK. Furthermore, the data type may be specified as "string" for the writer function. These strings may correspond messages corresponding to system or controller errors, run times, forecast information, etc. and serve as effective means of monitoring when coupled with PI Vision displays.

An additional "forecast writer" function takes the vector of future values and future timestamps generated from the forecast and writes the data back into PI AF, again using PI AF SDK in a loop to write to the future time stamps. The forecast is written to an attribute designated as future data within the PI System in order to write and overwrite data saved to future timestamps. Figure 4.2 gives the flow diagram of the control methodology.

`

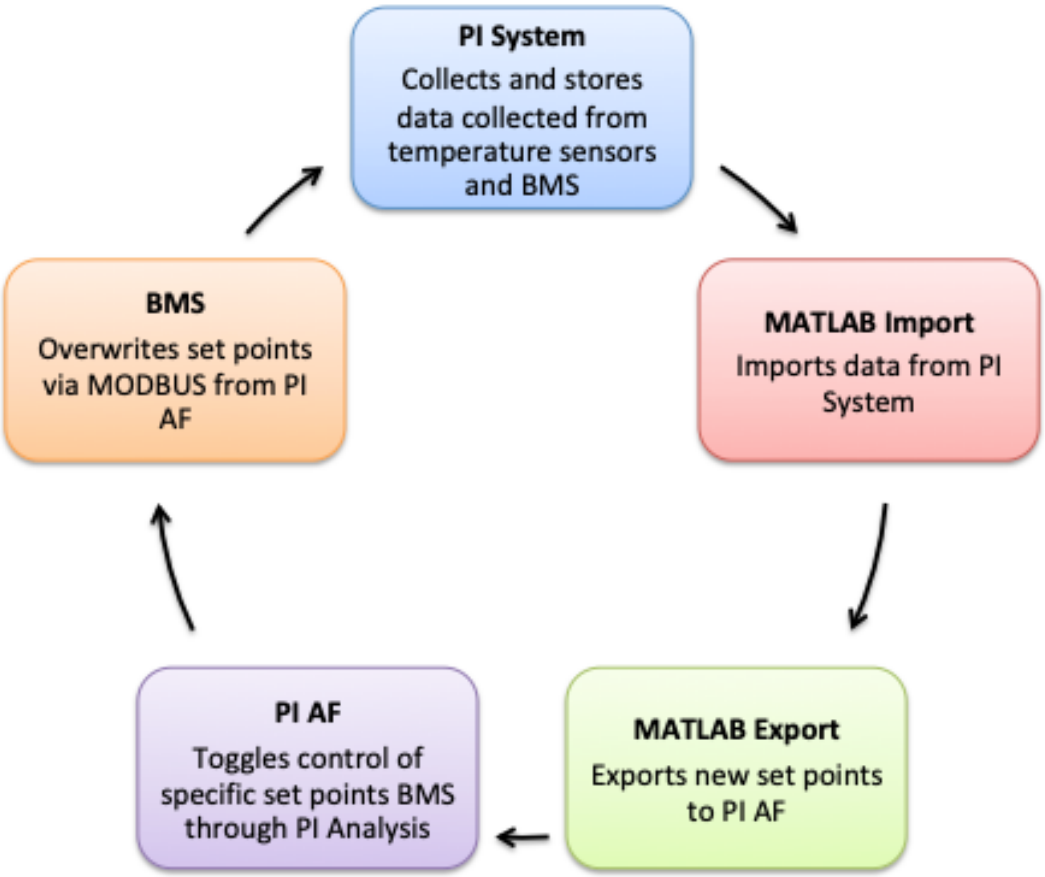

Figure 4.2 – Flow diagram of data within control methodology

4.3 PI Vision Displays for the Data Center Laboratory

For monitoring the conditions of the DCL, displays were made for the WTS modules and the CRAC units. PI Vision allows for data to be saved in collections, which creates identical displays, tables, etc. for multiple elements of the same type. Numerical data may also be color-coded based on user-defined limits to allow for more intuitive monitoring of the conditions of the data center. For instance, using this color-coding approach, users can define acceptable, warning, and critical ranges for temperature values. This is particularly useful for monitoring conditions that do not have remote

`

control capabilities, such that a data center operator can more effectively address problematic temperatures or data points.

The WTS modules have a display that shows the temperature value of each port, its corresponding forecast as a dashed line, status messages for the temperature and forecast, and metadata relating to the location of the sensor. The temperature status will change to “WARNING” if the temperature is within 5% of a prescribed temperature limit, and will change to “CRITICAL” if the temperature limit is exceeded. The forecast status will change if the forecast will exceed the temperature limit, and the status will give the time at which the temperature limit will be exceeded according to the forecast. The PI Vision display for the WTS units is shown in Figure 4.3.

`

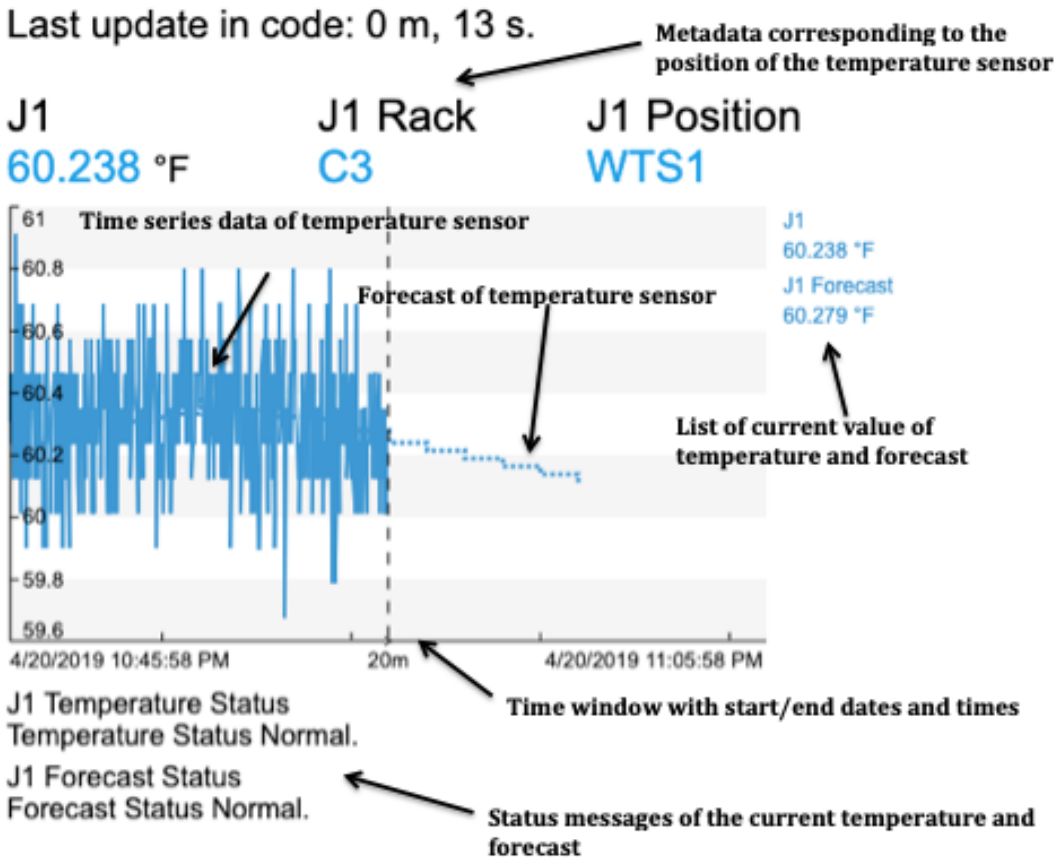

Figure 4.3 – PI Vision display of WTS Units, with time series data over the past half hour for the sensor temperature, a dotted forecast, metadata on the sensor location, and status messages

The PI Vision displays of the CRAC units show much of the data from the BMS, with certain data such as the return air and supply air temperatures, the temperature set point, etc. shown in time series graphs. The rest of the data is shown in tables. Putting the data for the CRAC units allows for effective monitoring of the conditions of the CRAC units while not necessarily needing access to the BMS itself. Figure 4.4 shows the time series graph for CRAC 2 displaying relevant temperature information of the CRAC unit.

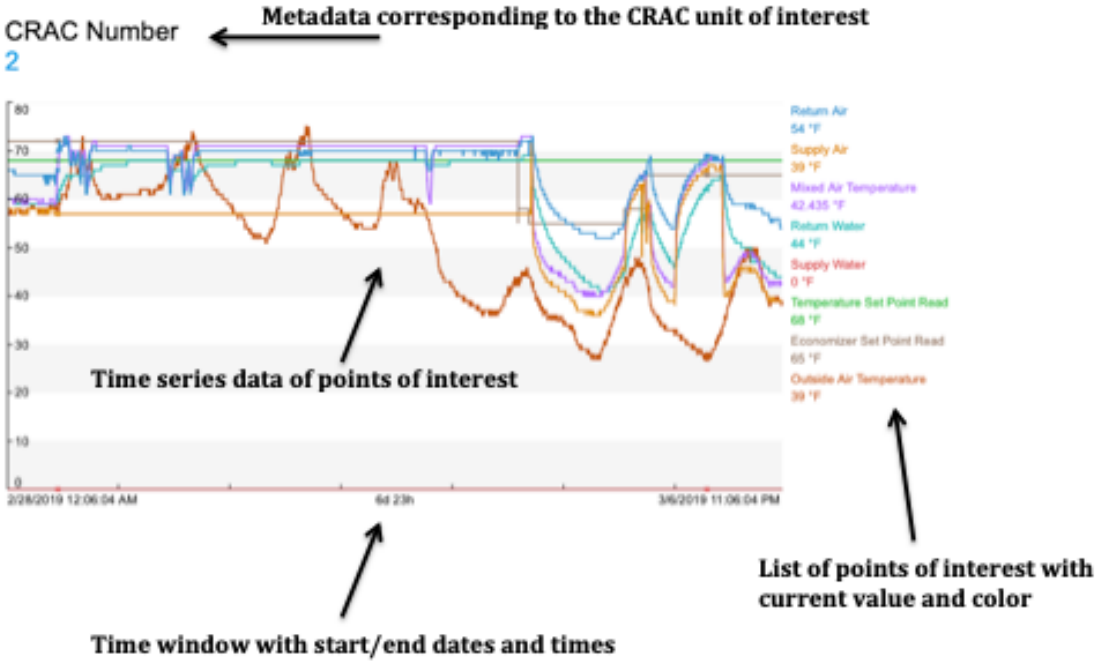

Figure 4.4 - Time series data of PI Vision display of CRAC 2, containing relevant air and water temperature measurements with accompanying supply air set point

`

CHAPTER 5. CASE STUDY 1 – CONTROLLING CRAC TEMPERATURE SET POINT TO OBTAIN TARGET SERVER INLET AIR TEMPERATURE

5.1 Motivation, Supporting Literature

Automatic control may be used to make temperature predictions in real time and adjust the settings of the cooling infrastructure. This results in reduced downtime: instead of a user analyzing the data to change the settings of the cooling infrastructure, a control method automatically applies the necessary changes, using active cooling to prevent over-provisioning. Therefore, it is the goal of this case study to devise a heuristic method that utilizes temperature prediction and controls to communicate with cooling infrastructure to more effectively cool the data center.

Boucher [11] ran multiple experiments focusing on the results of controlling different aspects of data center cooling equipment. When controlling the CRAC supply air temperature, keeping fan speeds and vent positions constant, results in a relatively linear relationship between the CRAC supply air temperature and the air inlet temperature. Furthermore Bash [12] defines a Thermal Correlation Index used to quantify the change in temperature readings at the rack level in response to a temperature range at the CRAC level. Finally, Li [6] utilizes a heuristic, auto-regressive model that relies on real-time environmental data to predict inlet and exhaust temperatures based on relationships derived from thermodynamics and past data trends for specific parameters.

`

Using this method, there were reasonably accurate temperature predictions for windows ranging from 5-20 minutes based on training data sets ranging from 15-90 minutes.

Based on these models, the temperature predictions are based on a linear relationship between the CRAC supply air temperature and the server inlet temperature from Boucher [11]. The TCI devised from Bash [12] represents the "slope" of the line of the server inlet temperatures measured by the WTS units vs. the CRAC supply air temperatures. However, the CRAC units in the DCL are based on the return air temperature. Therefore, the TCI in this case study are redefined to be the change in temperature of the server inlet temperatures divided by the change in temperature of the CRAC return air temperature set point.

$$TCI = \frac{\Delta T_{Rack}}{\Delta T_{RA,set}} \tag{5.1}$$

Based on Li's auto-regressive model [6], coupled with the linear relationship between supply air temperature and server inlet temperature from Boucher [11], it is the goal of this case study to devise heuristic, regression-based temperature prediction in conjunction with changing cooling infrastructure settings to reach a target rack inlet temperature. Furthermore, this method must constantly update with feedback in order to accurately reach the target temperature. While this approach has not been directly used in previous literature, there are other examples of similar concepts of using controls within the larger context of date centers.

The primary validation for this case study comes from Bash [12] and the use of TCI with the DSC controller, representing room-level control based on feedback from

`

rack inlet temperatures. This case study seeks to use a similar approach in using temperature measurements with controls to calculate a new temperature set point, however with the addition of using regression-based temperature prediction.

5.2 Calculating Thermal Correlation Index

The thermal correlation index was calculated by using an APC Server Thermal Simulator and CRAC 5, a downflow Liebert FH740CMAIES055 unit, in the experimental zone of the DCL illustrated in Figure 5.1. The return air set point of the downflow CRAC unit was increased by 0.55°C (1°F) over a temperature range of 18.3-29.4°C (65-85°F), and the temperature measured at the inlet of the server simulator was recorded after 10 minutes to allow for a relative stabilization. The server inlet temperature variation with the CRAC return air temperature was used to generate a linear fit for each J1-J4 port, and the average slope of the line was used to obtain the TCI. This was done over several trials, altering the heat load and fan speed of the server simulator. The manufacturer's absolute uncertainty of the temperature sensors is ±0.3°C [20], while the uncertainty of the return air temperature is ±1°C [21]. Carrying out these uncertainty values in an uncertainty analysis results in a relative TCI uncertainty of ~3.5% for the range of temperature values used for TCI calibration.

`

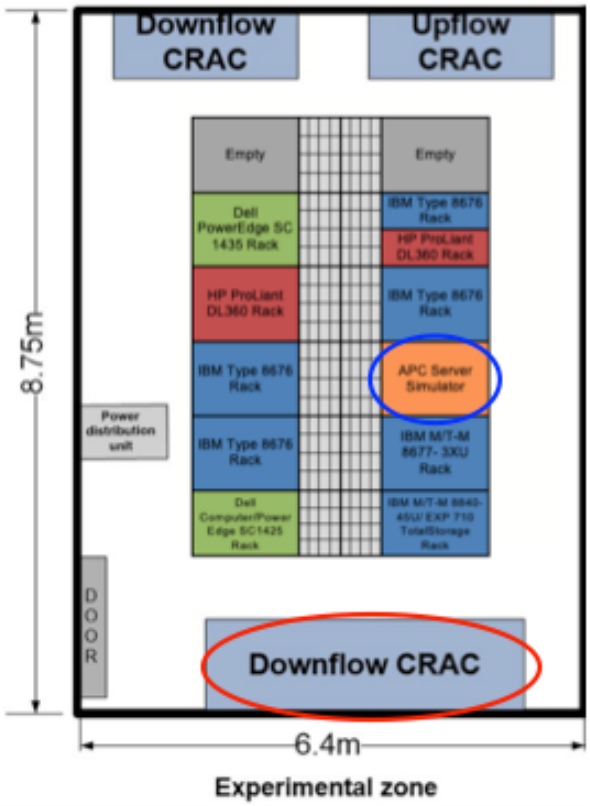

Figure 5.1 – CRAC unit and server simulator used for case study within DCL infrastructure

The results of the TCI calibration trials are shown in Table 5.1. It can be seen that the TCI for the majority of the trials was in the range of ~0.9-1, implying a slope of near unity. Trial 1a, illustrated in Figure 5.2, had the same conditions as Trial 1, however instead of increasing the temperature set point from 65°F to 85°F, the temperature was decreased from 85°F to 65°F in order to illustrate that the TCI is comparable for both heating and cooling. Trial 2 was incomplete due to the severe over-provisioning of the server simulator, and the upper temperatures were not tested so as not to damage the server simulator. Trials 6 and 7 feature a change in the heat load midway through the experiment, and their TCI values were measured for each server setting and for the

`

overall trial. For these trials, it did not matter whether the heat load was increased or decreased: the TCI was higher at the lower temperature ranges compared to the upper temperature ranges. However, the overall TCI values of Trials 6 and 7 were comparable to the other trials. Given the relative uncertainty of the TCI as 3.5% based on the absolute uncertainties given by the manufacturers, the typical TCI absolute uncertainty in these trials is in the range of 0.00315-0.0035.

Table 5.1 – TCI Calibration Results

Trial	Description	TCI
1	0 kW, 0 m^3/s	0.92
1a	0 kW, 0 m^3/s (85°F → 65°F)	0.91
2*	20 kW, 0.55 m^3/s	1.04
3	20 kW, 1.18 m^3/s	0.93
4	10 kW, 0.59 m^3/s	0.88
5	10 kW, 0.76 m^3/s	0.94
6	10 kW, 0.59 m^3/s →20 kW, 1.18 m^3/s	0.99
6a	10 kW, 0.59 m^3/s	1.18
6b	20 kW, 1.18 m^3/s	0.65
7	20 kW, 1.18 m^3/s→10 kW, 0.59 m^3/s	0.90
7a	20 kW, 1.18 m^3/s	1.23
7b	10 kW, 0.59 m^3/s	0.79

`

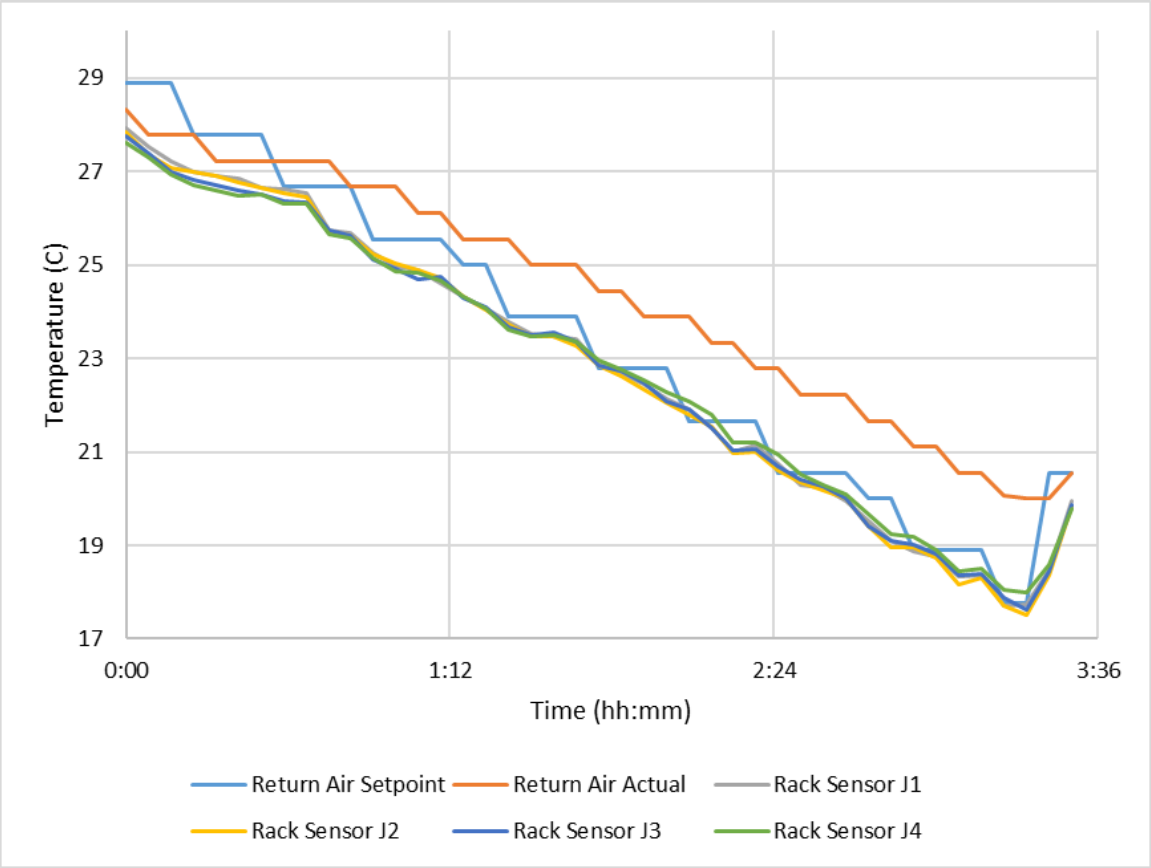

Figure 5.2 – TCI Calibration in Reverse

Figure 5.3 gives the values of the return air temperature set point, the actual return air temperature, and the values for WTS ports J1-J4 for TCI trial 4. There is some noticeable "bowing" at the temperature extremes: the return air temperature set point is higher than the actual return air temperature at the upper temperature extremes (>27°C set point) and lower than the actual return air temperature at the lower extreme (<20°C set point). This bowing phenomenon was present in all trials. Furthermore, the measured inlet air temperatures from WTS ports J1-J4 follow the shape of the actual return air temperature as opposed to the return air temperature set point. In this trial, the return air temperature "peaked" to match the set point, and then fell to a lower level. While this peaking behavior is unique to Trial 4, this illustrates that there is often some offset

`

between the return air temperature and its associated set point, likely due to the adjusting CRAC equipment settings.

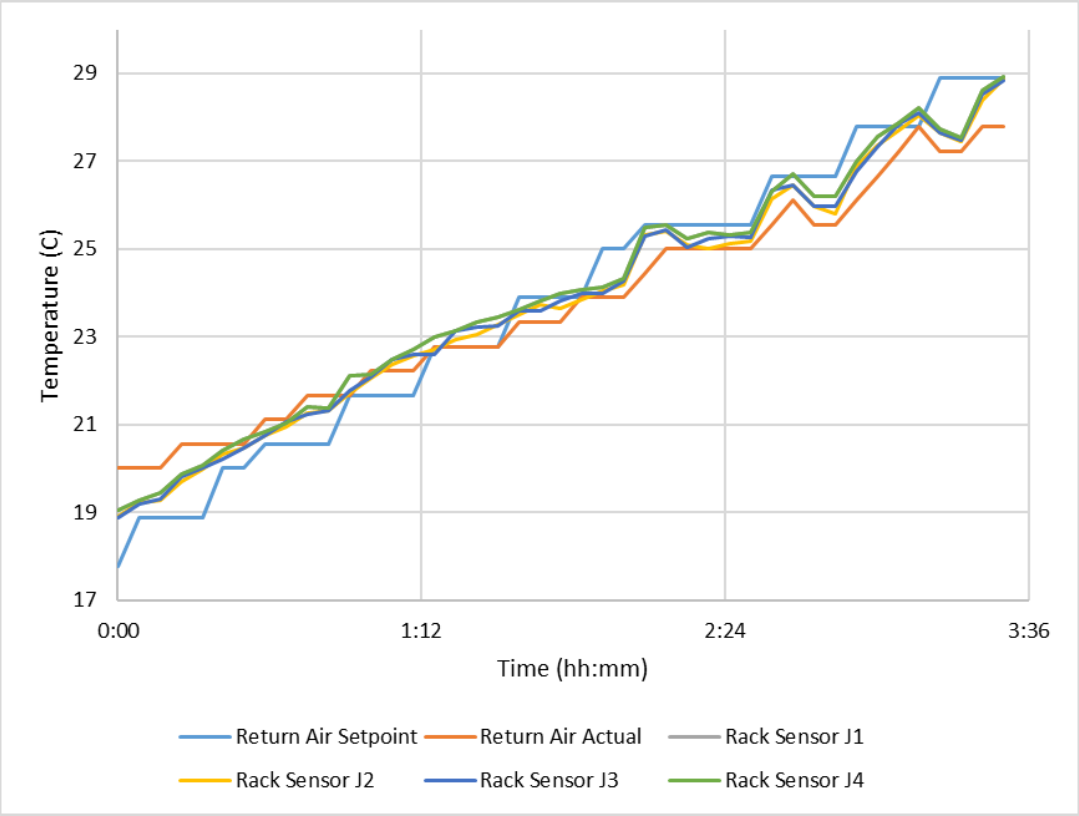

Figure 5.3 – Temperature values for TCI Trial 4

Despite the offset between the return air temperature and its associated set point and the bowing behavior at temperature extremes, the TCI trials validate the linear relationship between the supply air temperature and the server inlet air temperature proposed by Boucher [11]. The TCI metric accounts for the heat load within the data center, which was assumed constant as the heat dissipation from the individual racks, lighting, and cooling equipment did not vary significantly.

`

5.3 Coding in MATLAB and PI

The temperature values of ports J1-J4 of the WTS unit measuring the inlet air temperatures of the server simulator were imported into MATLAB using the "import" MATLAB function. Once the data was imported, a first-order fit was applied to a vector of past data to generate a linear forecast of future times. This forecast, along with conditional statements used to monitor the status of the temperature and the forecast, were then written back into the PI System using the "writer" MATLAB function. This import and export of data is put in a continuous loop which pauses every 30 seconds to get semi-real time data, as without the pause function the connection to the BMS typically drops.

As discussed earlier, Li used predictions for windows training data sets ranging from 15-90 minutes for look-ahead windows of 5-20 minutes [6]. Much in the same way, look-ahead windows for the linear fit of data were much less reliable, and as such the forecast lasted five minutes past the current time. However, given the linear fit of the forecast it was found that using larger training sets of data didn't necessarily result in the most accurate fits of data, especially since the larger training sets did not capture the immediate behavior of a temperature change in response to a change in the return air set point. Using smaller training sets typically reduced amplitude of oscillations yet increased the frequency, while larger training sets increase amplitude of oscillations yet decrease frequency. This phenomenon is illustrated in Figure 5.4, where the training set is set as 10 minutes, and Figure 5.5, where the training set was set as 2 minutes. With the 10-minute training set, the oscillations around the target temperature have higher amplitude, meaning there is a higher temperature difference around the target

`

temperature. Meanwhile, in the 2-minute training set the response features more rapid oscillations around the target temperature. As such, the training set was set to 5 minutes.

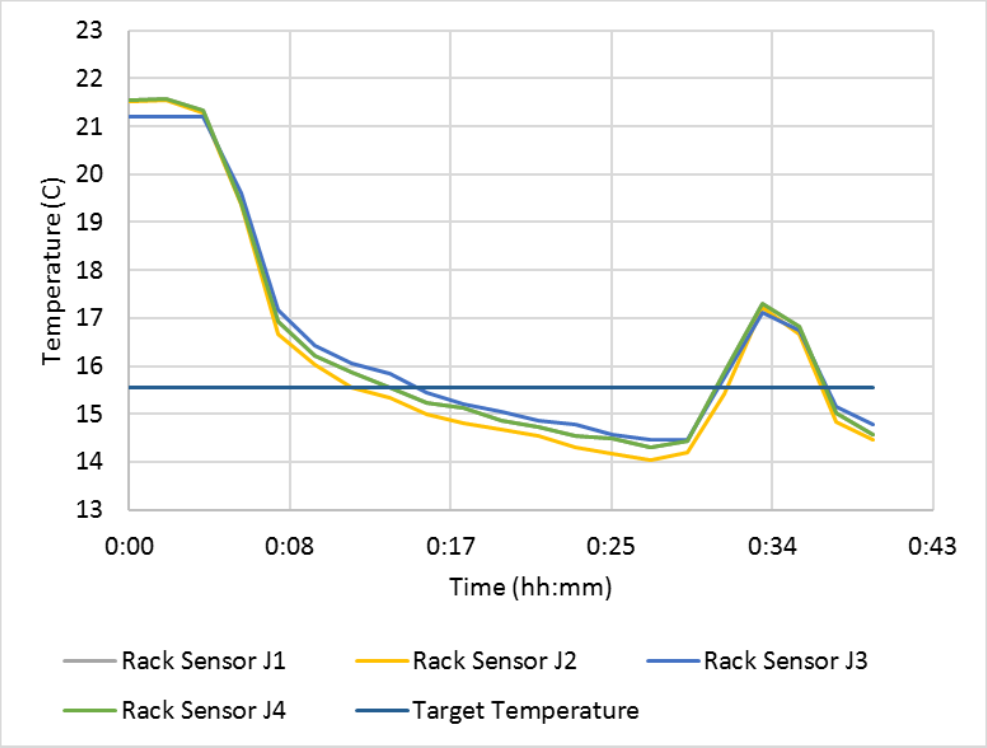

Figure 5.4 – 10-minute training set

`

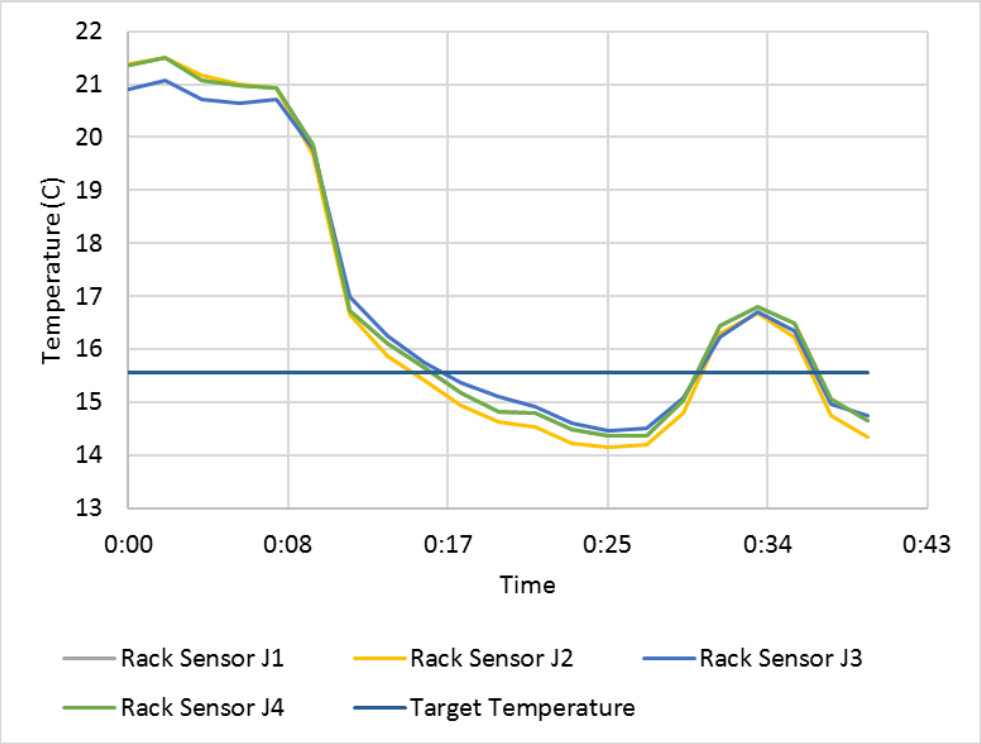

Figure 5.5 – 2-minute training set

Figure 5.6 displays a diagram of the inputs and outputs of the MATLAB control algorithm. $T_{CRAC,\ in}$ represents the current CRAC return air temperature, $T_{Rack,\ in}$ is the imported rack inlet temperatures, $T_{Rack,\ target}$ is the target rack temperature, $T_{CRAC,\ out}$ is the new CRAC return air temperature set point, and $T_{Rack,\ forecast}$ is the rack inlet temperature forecast.

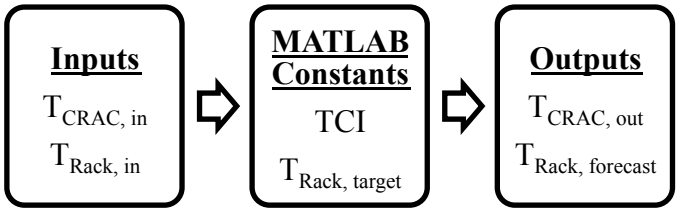

Figure 5.6 – Diagram of MATLAB control algorithm with associated variables

$T_{CRAC,\ out}$, given in Equation 5.2, is calculated by rearranging the rack and CRAC temperature values in the formula for TCI (Equation 5.1).

`

$$T_{CRAC,out} = \frac{T_{Rack,target} - T_{Rack}}{TCI} + T_{CRAC,in} \quad (5.2)$$

The value of T_{Rack} is based on both the current rack temperatures of $T_{Rack,\ in}$, as well as the final value of the forecast from $T_{Rack,\ forecast}$.

It was observed that having controls based solely on the forecast featured fast responses in changing the temperature. However, forecast control frequently resulted in non-converging oscillatory behavior, in which the forecast would predict an overshoot of the target temperature, and as a result would over-correct the value of $T_{CRAC,out}$. This phenomenon of using solely forecast-based control is illustrated in Figure 5.7.

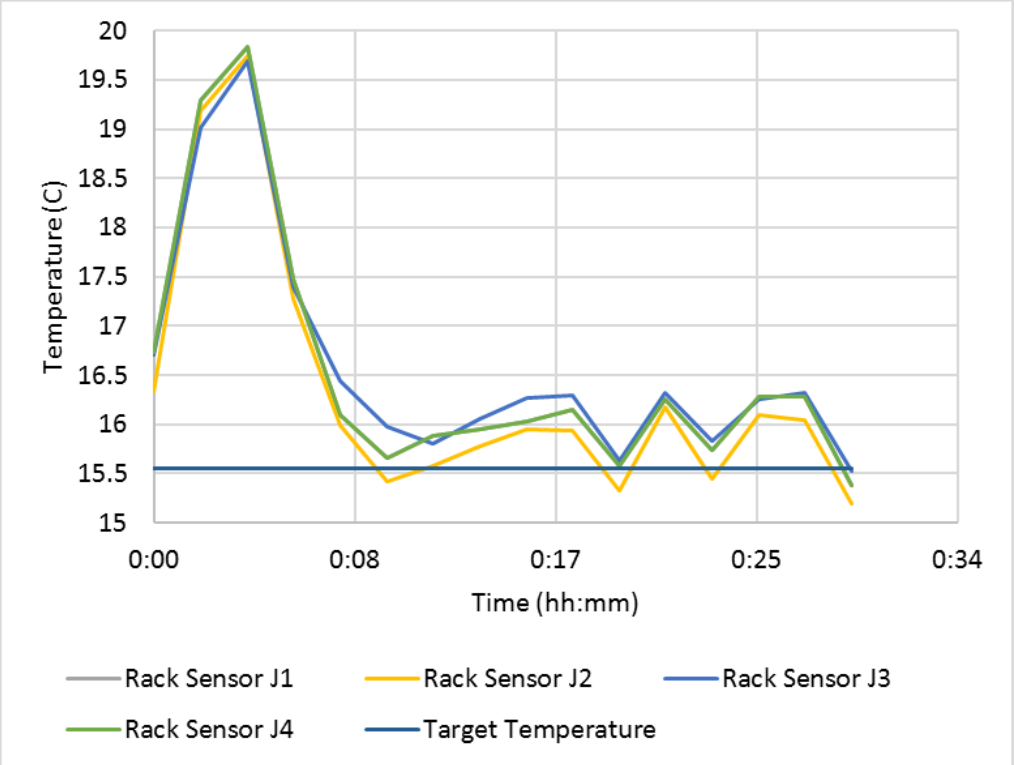

Figure 5.7 – Control methodology using forecast temperature only

`

Meanwhile, using only current rack temperature control takes a significant time to reach steady-state, particularly when the difference between the current temperature and the target temperature is higher than ~5°F. However using the current temperature results in improved steady state. This phenomenon of using solely the current temperature difference is illustrated in Figure 5.8. Using trial and error, forecast control was used if the current temperature was outside of ±2.78°C (±5°F), and within these bounds the current temperature was used as the value of T_{Rack}.

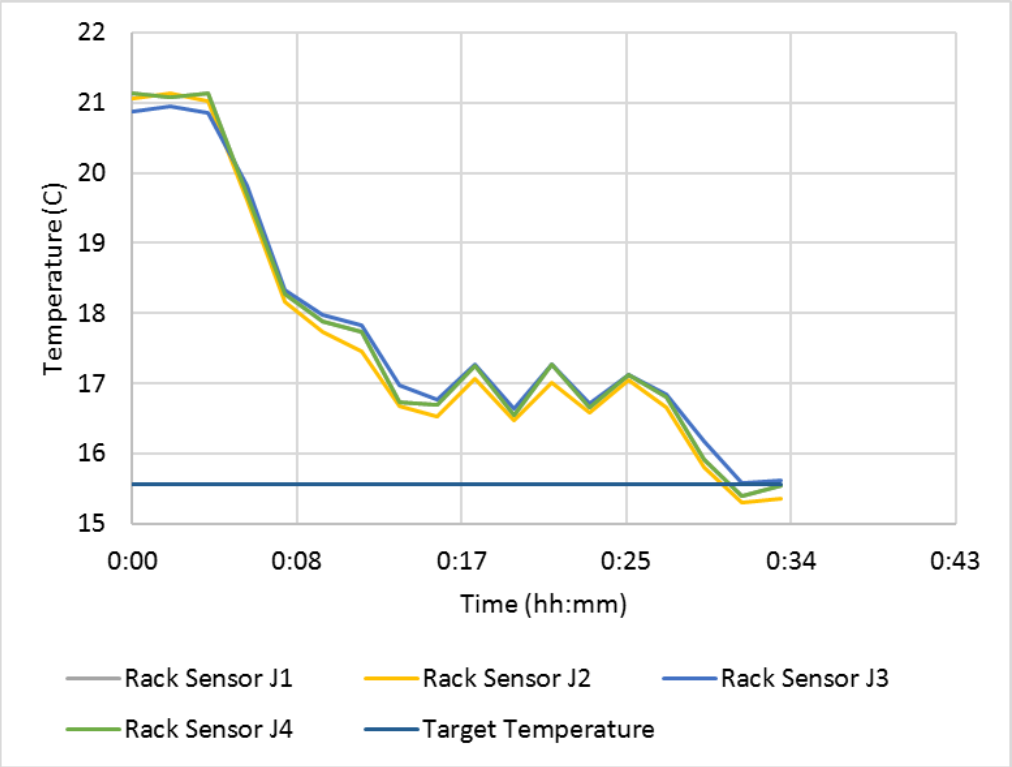

Figure 5.8 – Control methodology using current temperature only

When a new value for the CRAC return air temperature set point is written to the PI System, the MODBUS configuration allows for overwrites to the set point in the BMS. The time between changing the return air temperature set point in the PI System and observing the change in set point in the BMS was usually in the range of ~5-10 seconds.

`

Despite this, using this methodology one simply has to define the target temperature and run the code.

Figure 5.9 gives the overall process flow of the control algorithm. The data moves continuously between the PI System and MATLAB using PI AF SDK, and between the PI System and the BMS using MODBUS communication. The basis of control is the temperature difference between the rack temperature and the target temperature, and this iterative flow allows for constant adjustment of the CRAC settings to accurately reach the target temperature. Furthermore, using the forecasted rack temperature allows for more proactive temperature control as opposed to reactive control, although the use of the current rack temperature within the ±2.78°C temperature bounds does represent a more reactive control scheme. In this way, the control scheme is independent of the heat loads, pressure distribution, recirculation effects, etc. present within the data center and solely dependent on the rack inlet temperature and the CRAC return air temperature, allowing for rapid calculation.

`

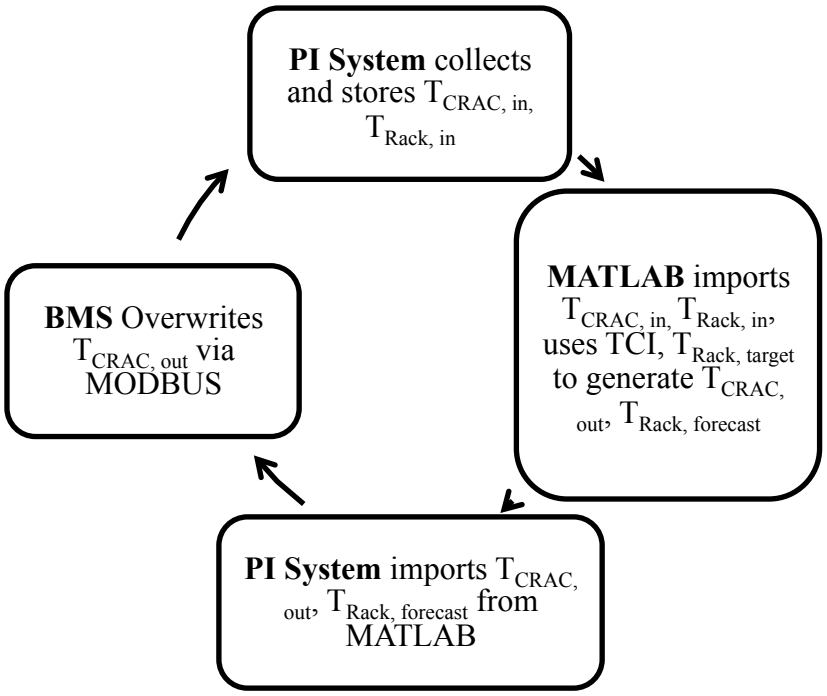

Figure 5.9 – Overall flow diagram of control algorithm, with specific variables associated with each step of the process

5.4 Results and Discussion

Using this MATLAB code, two trials were conducted to analyze the temperature evolution of the inlet air temperature in response to changes in the CRAC return air temperature set point. The first trial took the initial values of the temperature sensor at ~23°C (74°F), and set the target temperature to 20°C (68°F). The values from temperature sensors J1-J4, the return air temperature set point, and the actual return air temperature for this trial is given in Figure 5.10. The second trial featured initial values of the temperature sensor at ~20.5°C (69°F), and set the target temperature to 22.78°C (73°F). The values from temperature sensors J1-J4, the return air temperature set point, and the actual return air temperature for this trial is given in Figure 5.11.

`

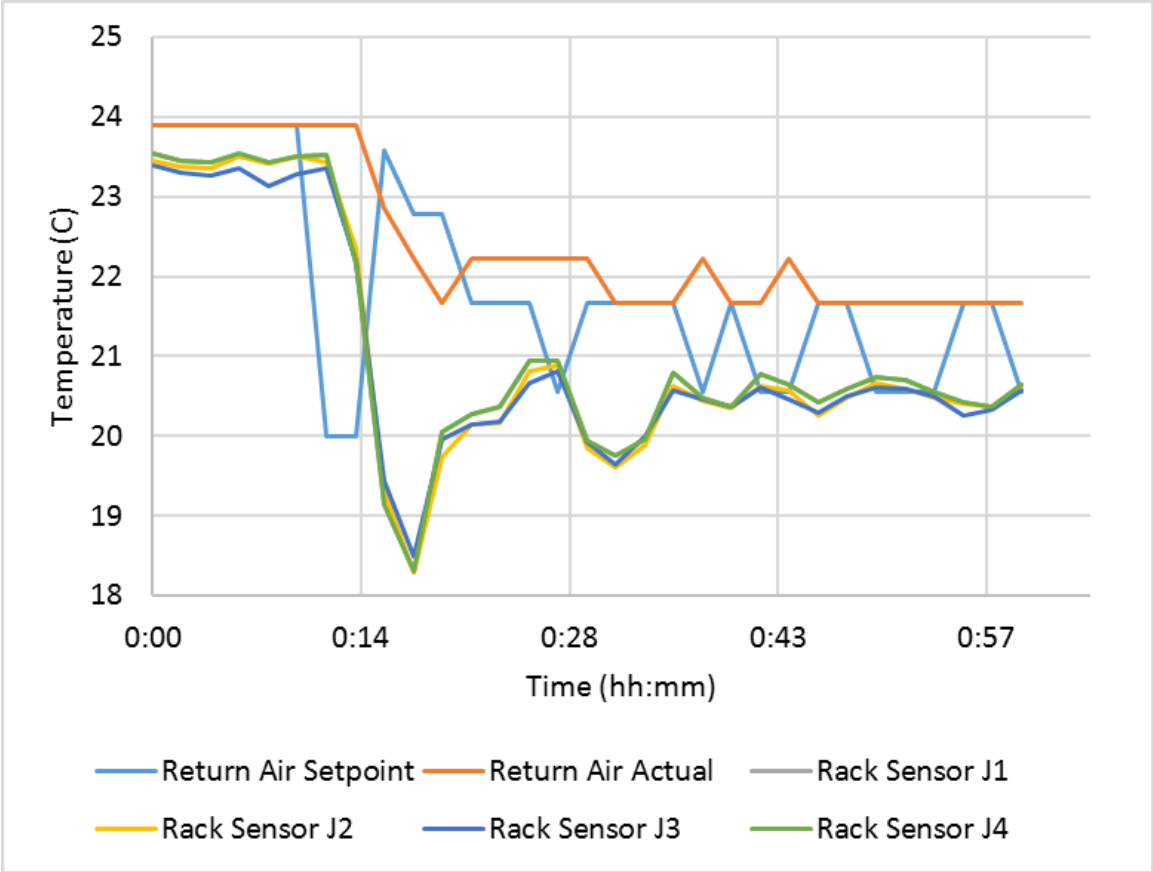

Figure 5.10 – First trial using control methodology setting target temperature to 20°C

The shape of the graph of the first trial resembles an underdamped step response. The time frame at which steady state is approached is approximately 30 minutes, with decreasing oscillations as time progresses. There is ~0.5°C offset between the steady-state value of 20.5°C and the target temperature of 20°C.

`

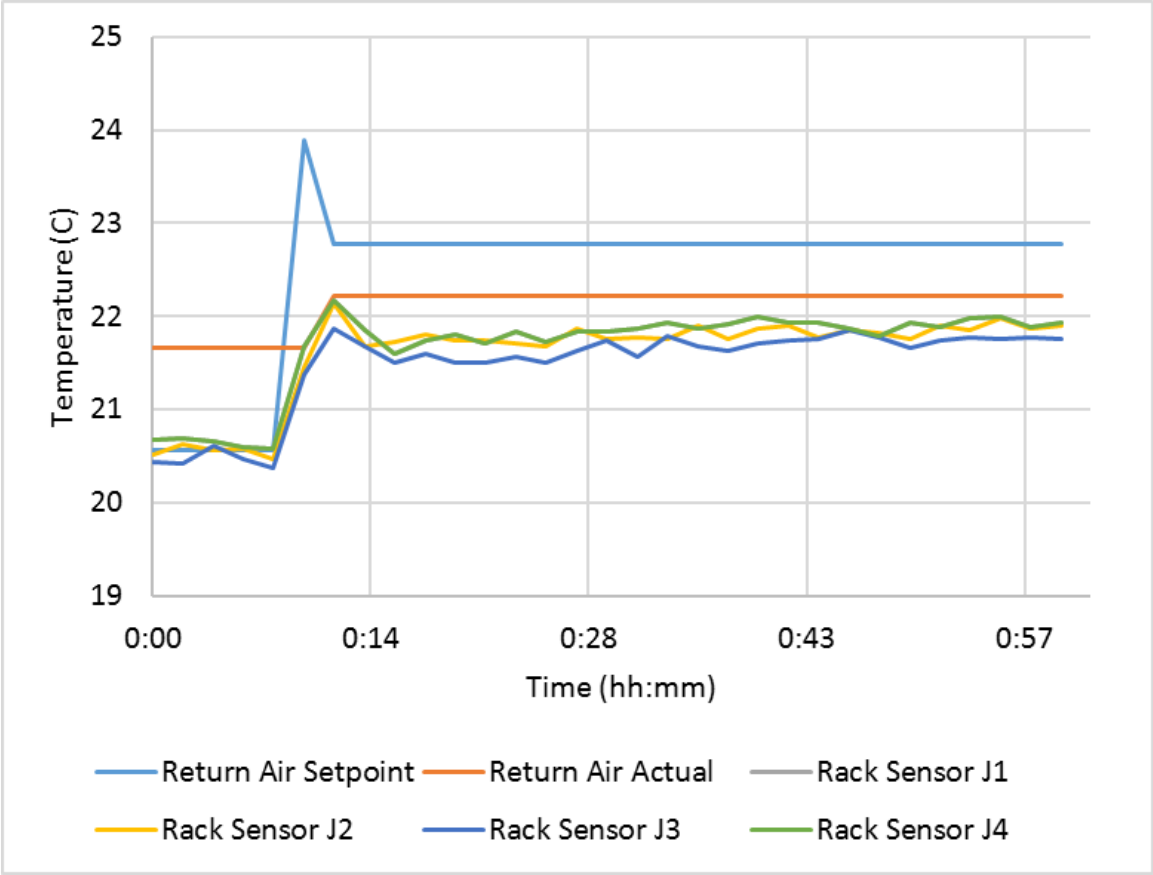

Figure 5.11 – Second trial using control methodology trial setting target temperature to 22.8°C

The shape of the graph of the second trial resembles an underdamped step response, similar to the first trial. Furthermore, a relative steady state is reached in the second trial at approximately 30 minutes, though there appears to be a marginal increase in the steady-state value as time evolves. The offset between the steady state for the second trial is slightly higher at ~0.7°C.

Table 5.2 gives the typical response times of instruments used in this control methodology. The MATLAB code that received all of the relevant data looped every 30 seconds in order to obtain the most recent data while still maintaining a connection to the BMS. However, the typical response time for the BMS instruments was typically in the

`

range of approximately one minute. Meanwhile, the responses of the WTS modules were typically one second. While this control frequency results in rapid generation of forecasts, the rate at which the new set points were written does not correspond to the typical rate at which the temperature set point changes.

Table 5.2 – Typical response times of control variables for Case Study 1

Variable	Average Response Time (mm:ss)	Minimum Response Time (mm:ss)
Return Air Temperature Set Point	~1:00	0:01
Actual Return Air Temperature	~1:00	0:04
WTS J1-J4	~0:01	0:01

It is important to note that the rate at which the return air temperature set point responded varied significantly. The minimum response time was in the range of seconds, meaning that this controller could react to the changes of the return air set point well within the 30-second window when the new temperature set point was calculated. By more carefully selecting scan classes for the BMS variables that are most critical to this case study, the response times of these variables would decrease to result in more dynamic control.

For both trials, the magnitude of change in return air set point is much higher than the magnitude of change of the actual return air temperature. This is due to the constant changing of the temperature set point such that there is some latency between changing the temperature set point and for the equipment of the CRAC to adjust the return air

`

temperature accordingly. While the TCI trials noted some difficulty in adjustment of temperature extremes when the set point was outside of the range of 20-27°C, the temperature set point for both trials remained outside of this range, and as such the effects from the bowing behavior from the TCI calibration was not evident.

Table 5.3 – Absolute uncertainties of the instruments used with the controller utilized in Case Study 1

Measurement	Uncertainty
Temperature sensors J1-J4	0.3°C (0.54°F) [20]
Return air temperature set point	1°C (1.8°F) [21]
Return air temperature sensor	0.36°C (0.65°F) [22][23]
BMS to PI temperature resolution	0.278°C (0.5°F)
Corrected return air temperature set point	1.04°C (1.87°F)
Corrected return air temperature sensor	0.45°C (0.81°F)

Table 5.3 gives a full list of the absolute uncertainties of each of the instruments used in the controller. The uncertainties of the wireless temperature sensors and the return air temperature set point were previously discussed in the TCI calculation, leading to a TCI uncertainty of roughly 3.5% based on the temperatures used. Additionally, one must consider the uncertainty of the actual return air temperature, used in Equation 5.2 as $T_{CRAC,\ in}$. Looking at the specification sheets of the DCL, the typical temperature sensors used are BAPI-BA/10K-3(11K)-D-2"-NB-10 temperature sensors [22], which have a

`

thermistor uncertainty of 0.2 °C (0.36°F) and a RTD uncertainty of 0.3°C (0.55°F) [23], giving a combined uncertainty of 0.36°C (0.65°F). Furthermore, one must also consider the resolution between the BMS and the PI System, as measurements are recorded to the nearest 1°F (0.55 °C). Therefore, the corrected return air temperature set point uncertainty is 1.04°C (1.87°F), and the corrected return air temperature uncertainty is 0.45°C (0.81°F).

While the controller does not account for the uncertainty of these variables, the trials still demonstrate a high degree of rack-level temperature control. The rack temperature sensors have a much lower uncertainty compared to the return air temperature values from the BMS. Given the effectiveness of this controller to reach a target temperature within a ±0.3°C band of uncertainty, this controller still is a highly effective tool for reaching and maintaining a user-defined target rack inlet air temperature. The uncertainty bands for the first trial are given in Figure 5.12.

`

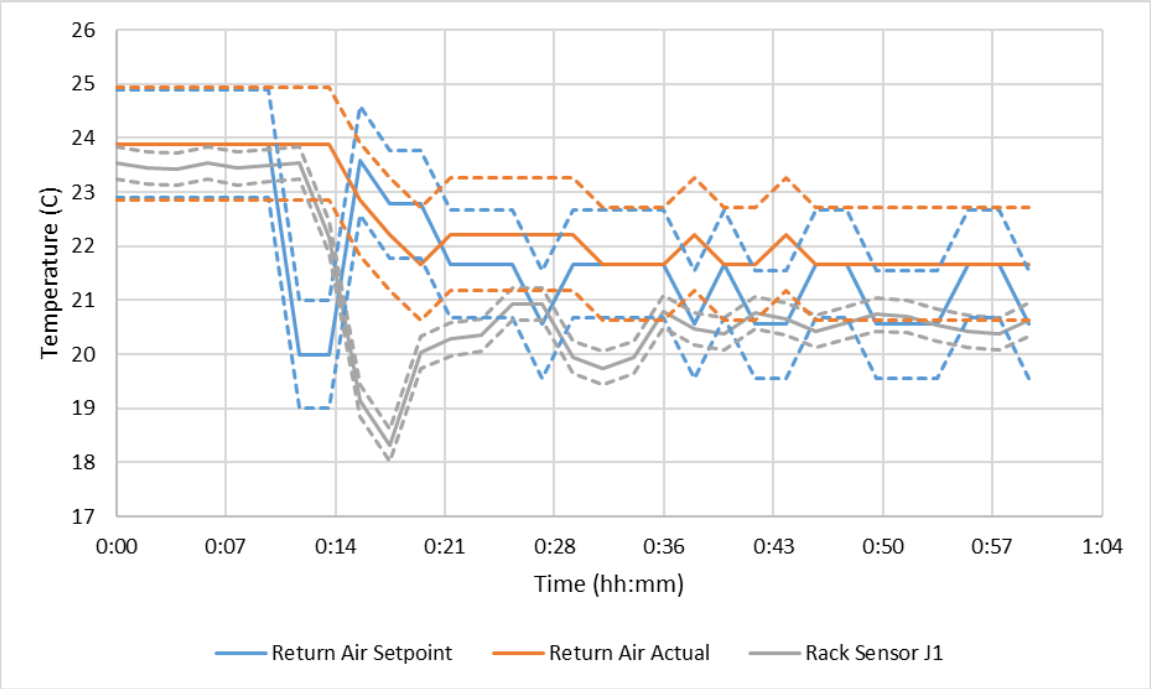

Figure 5.12 – Trial 1 return air temperature set point, return air temperature, and J1 rack temperature with uncertainty bounds given by dashed lines

These trials represent server air inlet temperature control based on a single rack. When observing the effects of this control on other server racks in the experimental section, the shapes of the air inlet temperatures responses of the other racks resembled that of the trials, however with higher or lower temperatures. This is due to the nature of the TCI being a static variable depending on the pairing of a specific CRAC unit to a specific rack location [12]. While the server simulator featured a TCI of ~0.95, other server racks may feature higher or lower TCI values that result in the higher or lower temperatures compared to the trials of the server simulator used in this study. The best way to address this would be to take a composite TCI and temperature forecast across the server racks of interest and use that as the basis of control. However, there may be an instance in which these composite values are insufficient for finer control of server inlet air on the server level. Given that this is a room-level study based on changing the

`

cooling infrastructure of the whole room, such finer control would be best addressed with localized cooling or adaptive tile vents. Such instruments could use local control based on the individual temperature forecasts and TCI values, while a composite temperature forecast and TCI are used on the room-level.

These trials demonstrate that controls based on temperature differences, either from forecasts or the current temperature, resemble a proportional controller. As time evolves the steady-state temperature approaches the target temperature for both trials. In this way, users simply have to define a target temperature and run the code, without the need to constantly monitor the conditions of the data center and having users change the settings of the cooling infrastructure.

Analogous to proportional-integral-derivative (PID) control, solely relying on proportional control resulted in an increased overshoot and settling time. The first trial featured a maximum overshoot of nearly 2.5°C, while the second trial did not overshoot the target temperature. Both trials also featured significant settling times of ~30 minutes to reach a relative steady-state temperature. Compared to the normal operation of the CRAC unit, it can be seen in these trials that changes in the temperature set point result in rapid changes to the return air temperature, and by extension the rack inlet air temperature in a range of ~2-5 minutes. However, the primary advantage of using this controller is that it allows in the first step of a more “hands-off” approach in monitoring the conditions of the data center. Instead of having an engineer having to constantly doing calculations to adjust the temperature set point, the code instead automatically adjusts the settings of the CRAC unit to approach the target server rack inlet temperature.

`

Coupled with more robust temperature prediction models, this controller represents an approach of smart cooling in the data center with user-defined functions.

Some of the shortcomings of the controller are due to limitations in reading and writing the data: running through one loop of the code takes a matter of milliseconds, however changing the return air temperature set point in the PI System and observing the change in set point in the BMS was observed to be in the range of ~5-10 seconds. Furthermore, one must also consider the transient effects of changing the temperature of the CRAC units. Since the code is limited to looping every 30 seconds, the controls are not based on truly real-time data, resulting in the oscillatory behavior exhibited by the code as the return air temperature adjustments overshoot. Finally, the data coming into PI from the BMS is cut off to the nearest degree Fahrenheit, which makes fine temperature control impossible, resulting in the typical offsets of steady-state temperature within ~1°C. Addressing these issues likely requires an improved BMS, with the ability to send and receive data from the PI System at a faster rate. However, these errors will persist using the current BMS infrastructure.

To address the issues with this method of control, it would be beneficial to utilize other aspects of PID control, namely integral and derivative control. Derivative and integral control could potentially be implemented by taking both the derivative and integral of the linear fit of the historical data, which could result in improved settling time and overshoot.

`

5.5 Summary

The need for supply air temperature control within data centers is becoming increasingly important to reduce the energy consumption of data centers. This case study presents an effective temperature control methodology using data gathered using the PI System and utilizing MATLAB for applying analysis and conditionals. This case study finds a linear relationship between the CRAC return air temperature and the rack inlet air temperature, and uses dynamic controls based on current and forecasted temperature differences to calculate optimal temperature set points. The result is a proportional controller that approaches a target temperature in a time frame of approximately 30 minutes.

`

CHAPTER 6. CASE STUDY 2 –CONTROLLING ECONOMIZER DAMPER POSITION TO MAINTAIN TARGET SPACE CONDITIONS

6.1 Motivation, Supporting Literature

As discussed in Chapter 1, ASHRAE maintains the recommended and allowable range of temperatures and humidities in a typical data center environment [2]. When utilizing economization the acceptable range of dry-bulb temperatures, the maximum dew point, and the limit for relative humidity for a typical data center typically increase. This shift in the acceptable range of conditions when using economization is the result of using warmer, moist air from the environment than typical air supplied from CRAC units. While some server manufacturers specify an operating range wider than the ASHRAE guidelines (such as those in classes B and C from Table 1.1), increasing the temperature past the limit results in higher power consumption from the servers and server fans. Additionally, the increased use of server fans will increase the noise load of the data centers, potentially resulting in the need for hearing protection equipment, which is costly to maintain. Finally, operating at humidity levels higher than the upper ASHRAE limit could result in degradation of printed wire boards, tape, and disk drives of the server, while operating below the lower limit raises risk for electrostatic discharge. Therefore, the benefits of energy savings from using economization must be weighed against the potential concerns of operating at a typically higher temperature and relative humidity [2].

`

Based on the limits of the different ASHRAE classes, it would be desirable to utilize the economizer whenever the outside air temperature is within these limits so as to reduce the demand for energy when cooling the space. However, this must also be weighed against any potential losses from bringing in air that is too warm and would require additional energy to cool, or air that is too cold that it goes below the temperature set point prescribed by the CRAC unit. Therefore, this case study seeks to propose an economizer controller that defines the temperature and humidity limits based on the ASHRAE classes, and dynamically controls the economizer to utilize the outside air when appropriate.

6.2 Methodology

This case study focused on the economizer settings of CRAC 2 in the DCL, which is a Liebert model FH529CVAG00 downflow unit. The dependent variable for this experiment is the mixed air temperature, while the independent variable is the outside air damper position.

The economizer is enabled when the outside air temperature is below the threshold given by the "OA Econ Cut Out Setpoint" minus the "OA Econ Cut Out Differential" in the BMS. Once the economizer is enabled, a user can set the outside air dampers to override, in which the user writes the damper percent as a set point. Otherwise, the dampers are set to an auto mode in which the dampers adjust automatically based on the sensed return air temperature, using the economizer when the outside air is cooler than the return air.

`

Using a similar approach as the first case study, data on the outside air temperature and humidity, along with the supply, return, and mixed air temperatures of the CRAC unit, as well as the space temperature and humidity may be imported into MATLAB using PI AF SDK. Conditionals are applied to calculate new outside air damper set points, and this value may be written back into PI AF using PI AF SDK, resulting in an overwrite in the BMS via MODBUS. Putting this process in a continuous loop allows the data to update dynamically.

This relation is based on the observation that the mixed air temperature is roughly a sum of the outside air temperature multiplied by the outside air damper percent, added to the return air damper multiplied by the return air damper percent. Since the sum of the outside air damper percent and the return air damper percent must equal 100%, this relation may be arranged according to Equation 6.1.

$$T_{MA} = T_{OA}\left(\frac{Damper\%}{100}\right) + T_{RA}\left(1 - \frac{Damper\%}{100}\right) \tag{6.1}$$

Therefore, solving for the outside air damper percent and choosing a target mixed air temperature, the governing relationship for calculating the outside air damper percent is given in Equation 6.2.

$$Damper\% = \frac{100(T_{MA,target} - T_{RA})}{(T_{OA} - T_{RA})} \tag{6.2}$$

The target mixed air temperature should be chosen to reduce the change in temperature across the cooling coil and variable speed drive (VSD) fan, as less cooling power required by these instruments would reduce the overall power consumption of the CRAC

`

unit. Therefore, the target mixed air temperature should be set close to the supply air temperature. From observation, a mixed air target of the supply air temperature + 1.11°C (2°F) resulted in minimal cooling power required.

In addition to controlling the outside air damper during normal operation, this code also reads the temperature and humidity data of the outside air and compares it to the space. If the space temperature or humidity is within 5% of the upper limit and the outside air temperature/humidity is higher than the mixed air temperature/humidity, the outside air damper will be set to 0%. Similarly, if the space temperature/humidity is within 5% of the lower temperature/humidity limit and the outside air temperature/humidity is lower than the mixed air temperature/humidity, the outside air damper will be set to 0%. Users define which class of data center space they wish to maintain (i.e. A1-A4) and the MATLAB code uses a "switch" case to define the upper and lower temperature and humidity limits accordingly. In this way, the data center envelope is within the limits prescribed by ASHRAE, with additional capabilities to be defined under user-defined temperature limits.

Similar to the first case study, the heat load within the data center was assumed constant as the heat dissipation from the individual racks, lighting, and cooling equipment did not vary significantly.

6.3 Results and Discussion

The first trial of economizer use is shown in Figure 6.1. This trial set the ASHRAE class to A3 while the supply air set point was maintained at 18.33°C (65°F).

`

Based on the control of the mixed air target near the supply air, the space, supply air, and return air values do not change much.

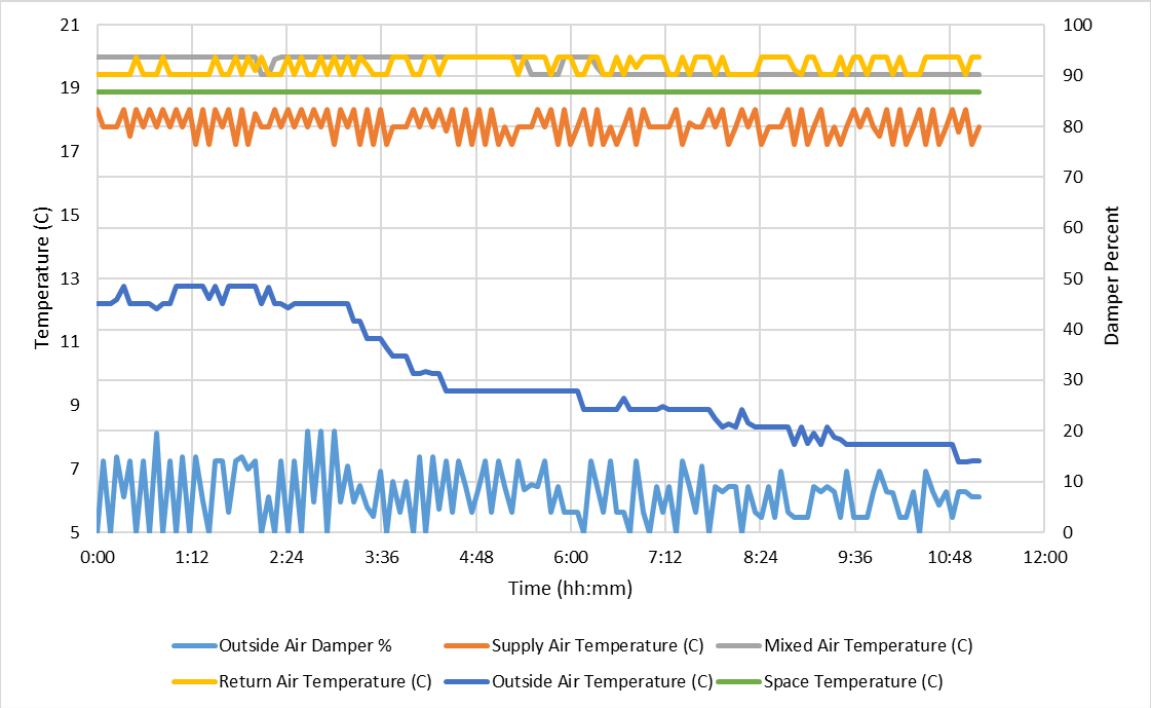

Figure 6.1 – Economizer control with supply air set point = 65°F, ASHRAE Class A3

The second trial of economizer control is shown in Figure 6.2. The supply air temperature set point was set as 12.78°C (55°F), and as a result there is a higher utilization of the economizer with 100% damper position in many instances. This data was recorded over a day/night cycle, in which there are high temperatures during the day and low temperatures at night.

`

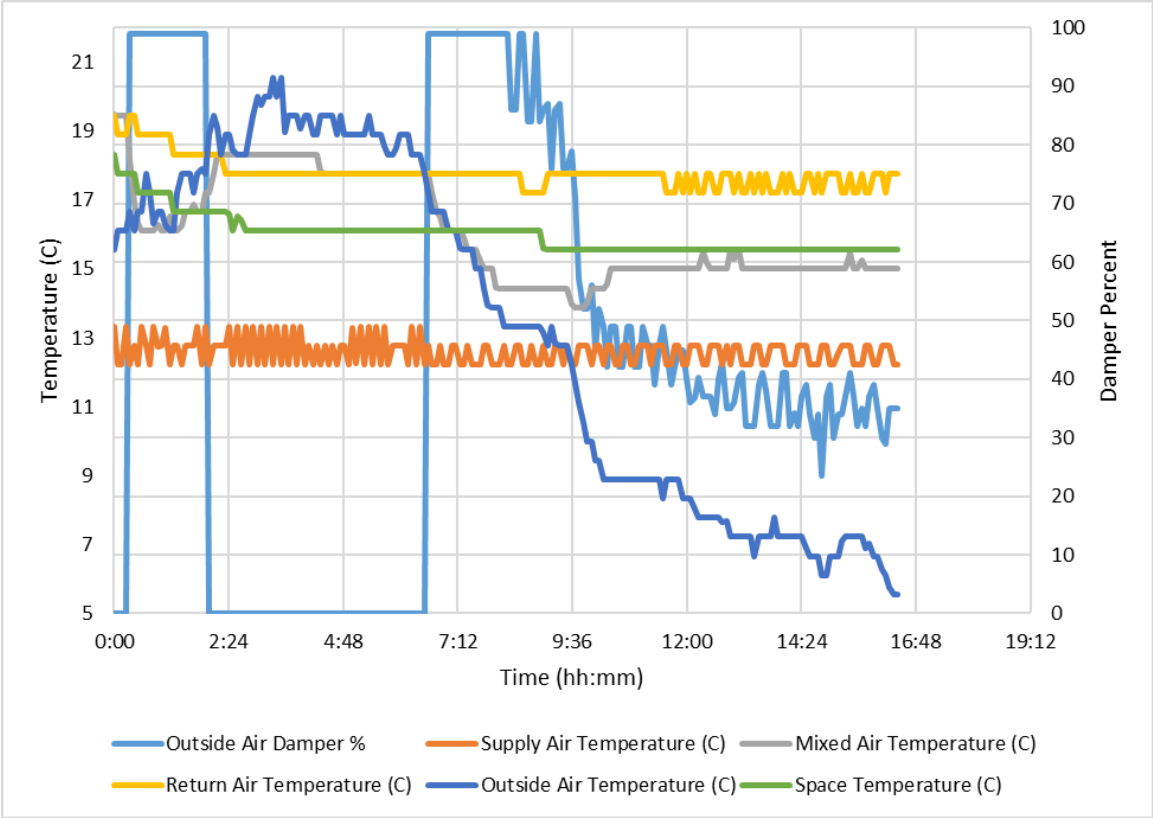

Figure 6.2 – Economizer control with supply air set point = 55°F, ASHRAE Class A3

The third trial of economizer control is illustrated in Figure 6.3. The ASHRAE class is changed to A2 and the supply air set point is 15.56°C (60°F). This trial began with the temperatures at a lower value due to leaving the economizer open over a longer period of time.

`

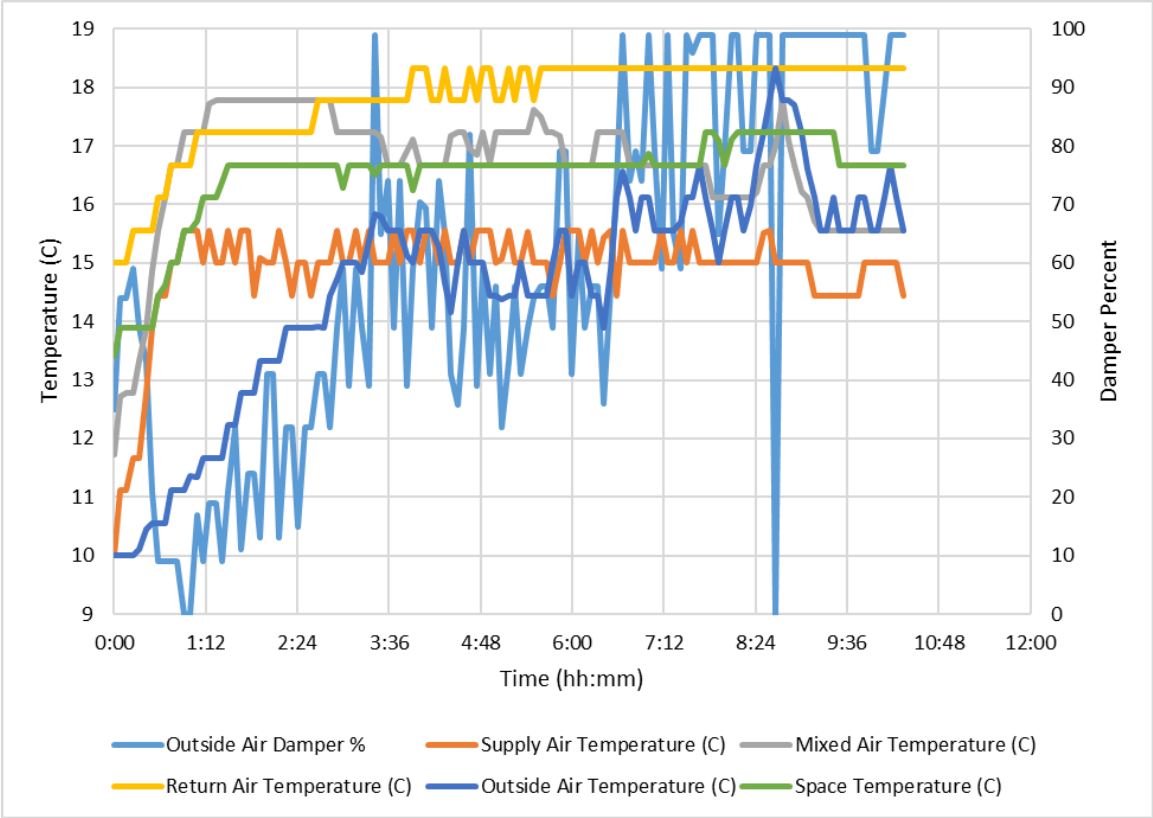

Figure 6.3 – Economizer control with supply air set point = 60°F, ASHRAE Class A2

Keeping the target mixed air temperature near the supply air set point results in less variation of the space temperature. In this way, the economizer damper will be utilized while maintaining the mixed air temperature. The result is finer temperature control of the space.

Maintaining this target mixed air temperature results in slightly different behavior than traditional economizer operation, in which the economizer is utilized when the outside air is cooler than the return air. The wide ranges of acceptable temperatures for ASHRAE classes A3 and A4 illustrate the normal phenomenon in which the economizer is utilized with less regard to the supply air temperature. In Trials 2 and 3, once the

`

outside air temperature exceeds the return air temperature, the economizer closes, as it would require more energy to cool the warmer air from the outside. Meanwhile, in Trial 1 the economizer is utilized to a lesser degree in order to maintain the relatively higher supply air set point of 65°F, and in Trial 3 the economizer is initially utilized less to adjust the supply air to its set point. In this way, the primary directive of the controller is to maintain the supply air near its set point, using the economizer when it is energy-efficient to do so. As a result, the variation of the space temperature was typically within 2°C.

The humidity values for these trials of economizer control are given in the appendix (Figures A.1 – A.3). While these trials featured humidity values within the limits of the A2 and A3 classes given in Table 1.1, if these limits were exceeded by the space and utilizing outside air humidities that would result in the violation of these space limits, there would be an automatic shut-off of the economizer, as this economizer does not feature humidification or dehumidification capabilities.

Table 6.1 gives a list of the absolute uncertainties associated with the controller used in this case study. The uncertainty of the supply air temperature set point comes from the specifications of the Liebert FH529CVAG00 manual [21], the temperature sensors are based on a typical BAPI-BA/10K-3(11K)-D-2"-NB-10 temperature sensor [22], with a thermistor uncertainty of 0.2 °C (0.36°F) and a RTD uncertainty of 0.3°C (0.55°F) [23], giving a combined uncertainty of 0.36°C (0.65°F). The resolution between the BMS and the PI System, as measurements are recorded to the nearest unit: 1°F (0.55 °C) for temperature values, 1% for relative humidity and damper values. Therefore, the corrected supply air temperature set point uncertainty is 1.04°C (1.87°F), the corrected air

`

temperature uncertainty is 0.45°C (0.81°F), the corrected relative humidity uncertainty is 3.04% and the corrected damper uncertainty is 0.71%.

Table 6.1 – Absolute uncertainties of the instruments used with the controller utilized in Case Study 2

Measurement	Uncertainty
Supply air temperature set point	1°C (1.8°F) [21]
Air temperature sensor	0.36°C (0.65°F) [22][23]
Air relative humidity sensor	3% [22][24]
BMS damper reading	0.5%
BMS to PI temperature resolution	0.278°C (0.5°F)
BMS to PI relative humidity resolution	0.5%
BMS to PI damper resolution	0.5%
Corrected supply air temperature set point	1.04°C (1.87°F)
Corrected air temperature sensor	0.45°C (0.81°F)
Corrected air relative humidity sensor	3.04%
Corrected damper reading	0.71%

While the controller does not account for the uncertainty of these variables, the trials still demonstrate a high degree of temperature control of the space while using the

`

controller. The variable with the highest degree of uncertainty, the supply air temperature set point, was kept constant. Seeing as the variation of the space temperature was typically less 2°C when using this controller, adding in the uncertainty of this measurement still results in variation of the space temperature < 5°C, which is still an acceptable degree of control given the ASHRAE class standards. Furthermore, closing the economizer whenever the space is within 5% of the upper or lower limits further illustrates the effectiveness of the controller.

6.3.1 Energy Consumption

The cooling power consumption of the data center can be broken down into three components: the power consumed by the chiller, the power consumed by the CRAC supply fan, and the power consumed by the server fans. Given this study is concerned with the conditions of the CRAC and the space, the server-level power consumption is considered constant.

The power consumption of the CRAC 2 supply fan was calculated by taking the maximum power consumption of 5.59 kW [21] and multiplying by the two fans and the VSD fan percentage from the BMS. As the fan speed was kept at a constant 30% for all of the trials, the fan power consumption remained a constant 3.354 kW.

The power consumed by the chiller may be calculated by calculating the heat removed by the CRAC and dividing by the coefficient of performance (COP), the ratio of cooling power to the work done by the chiller, shown in Equation 6.3 [25].

$$P_{chiller} = \frac{\dot{Q}}{COP} \tag{6.3}$$

`

The COP for the DCL was chosen as 3.3 for the chiller in the given ambient temperature range, Trane Series R Rotary Chiller RTAA130GYT01A [26]. Performing a simple energy balance on the air may approximate the heat consumption by the CRAC unit using Equation 6.4. [25]

$$\dot{Q} = \dot{m} C_p (T_{MA} - T_{SA}) \tag{6.4}$$

To calculate the volumetric flow rate of the air, the maximum flow speed of 21070 m^3/h [21] was multiplied by two for the two blowers of the specific CRAC 2 model, and then multiplied by the VSD fan percentage from the BMS. From there, the mass flow rate of the air in Equation 6.4 was calculated by taking these volumetric flow rate values and multiplying by a fixed density value of air chosen as 1.2 kg/m^3 for the given range of supply and mixed air temperatures, and then multiplying these values by two given that CRAC 2 features two of the blowers in parallel. With these calculated mass flow rates, along with the constant specific heat value of 1.006 kJ/(kg-K), and the supply and mixed air temperatures, the power consumption of the CRAC unit was calculated for each recorded value in the three trials of economizer use.

The power consumption for the first trial of economizer use is shown in Figure 6.4. The energy consumption generally follows the oscillations of the damper, with some lag as the temperatures adjust according to the damper positions.

`

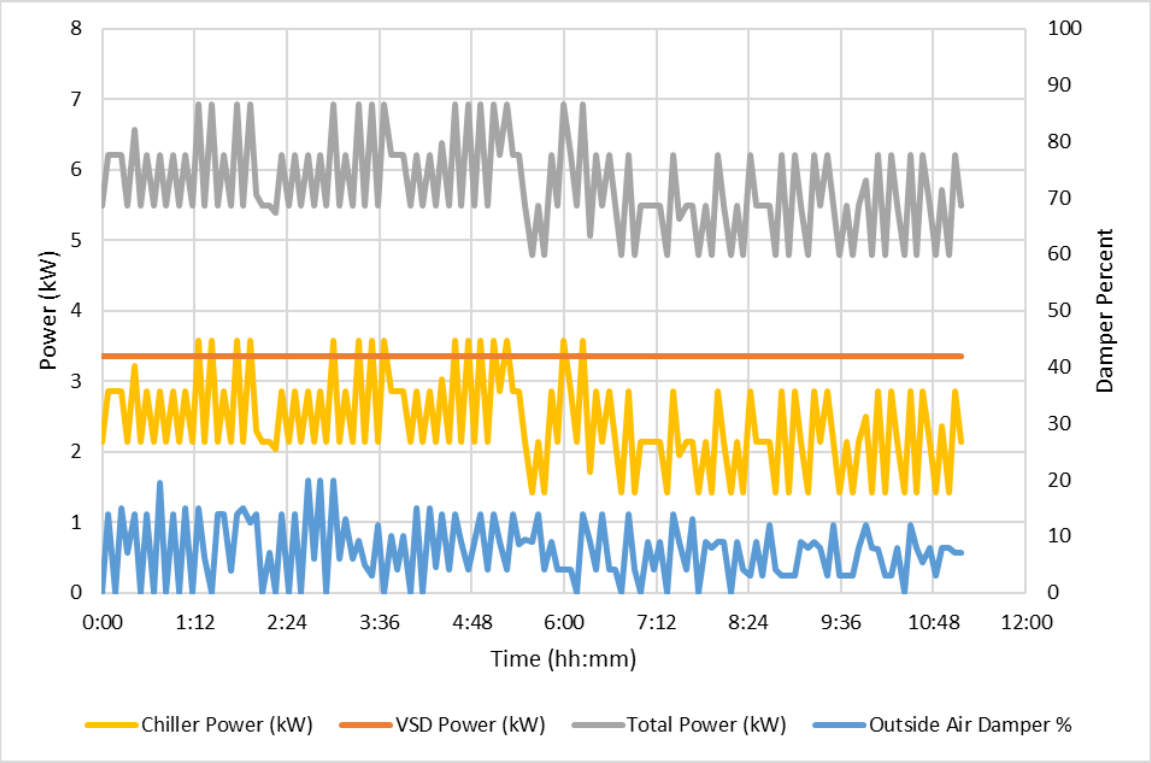

Figure 6.4 – Economizer power consumption with supply air set point = 65°F, ASHRAE Class A3

The power consumption of the second trial of economizer control is shown in Figure 6.5. This trial illustrates the behavior of the power consumption using the economizer much more clearly. This section where the economizer drops to zero occurred due to the outside air being too high to maintain the supply air temperature, and as a result all cooling was based around the return air. When the economizer was not in use, the power consumption rose from ~8 kW to ~11 kW. Additionally, when the outside air fell to below the return air and the economizer was put to use, there was an immediate drop in the power consumption, falling to ~6 kW in the night.

`

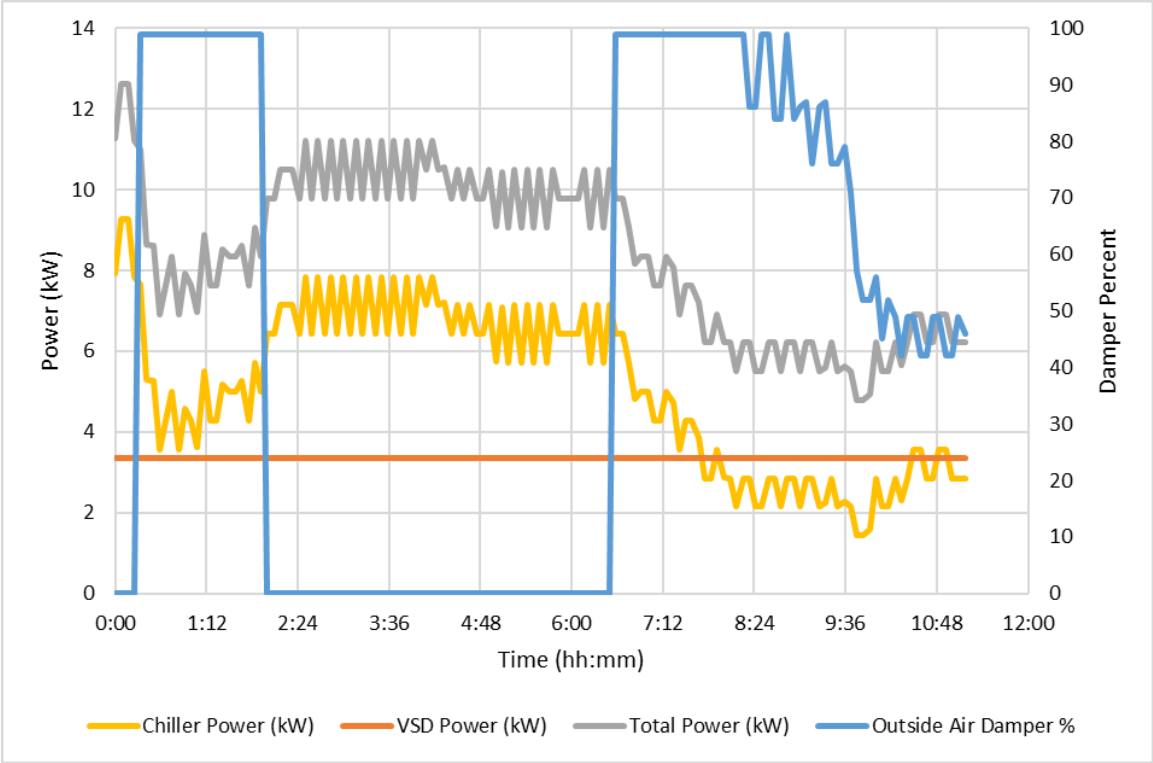

Figure 6.5 – Economizer power consumption with supply air set point = 55°F, ASHRAE Class A3

The power consumption of the third trial of economizer control is shown in Figure 6.6. Despite initial lower temperatures for the mixed and supply air, the controller's adjustment to obtain a supply air temperature value closer to the 60°F set point resulted in a large power consumption of ~6-7 kW as the mixed air over-corrected to adjust. Afterward, the mixed air stabilizes and the economizer is used more, resulting in a drop in energy consumption to ~5 kW. This illustrates a drawback of using this controller in the sense that the mixed air can over-correct to adjust to the correct supply air temperature, resulting in higher power consumption.

`

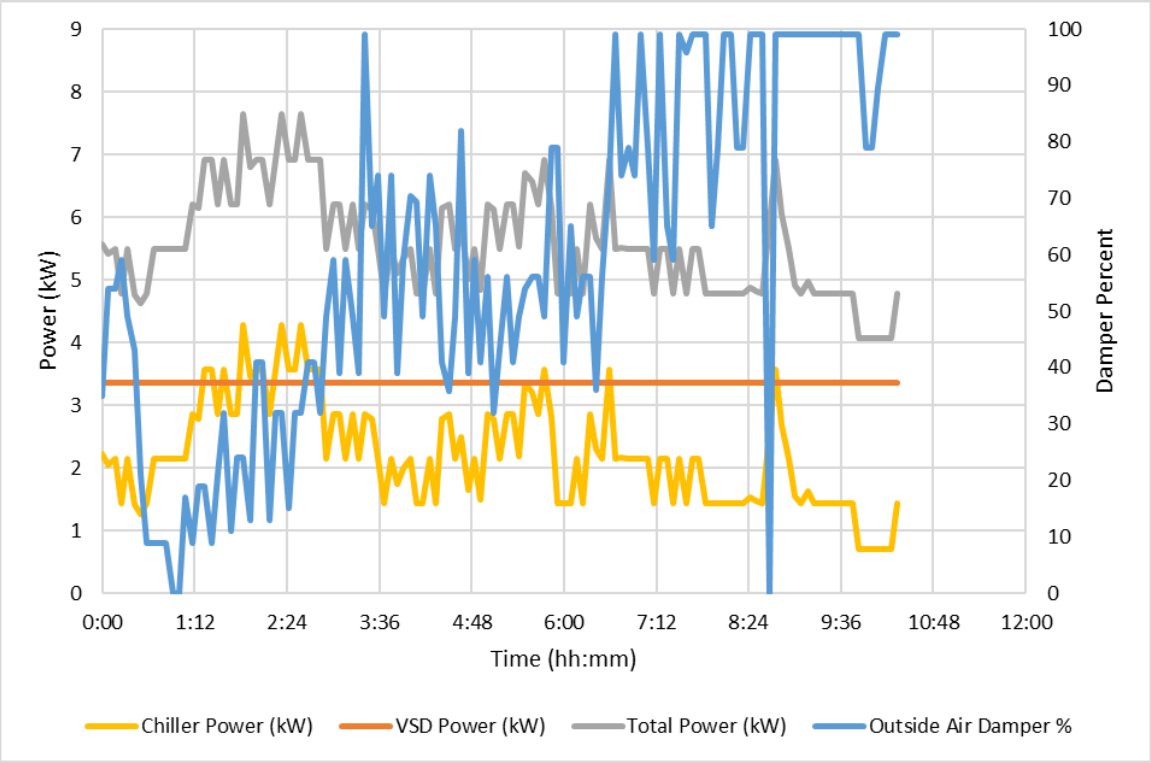

Figure 6.6 – Economizer power consumption with supply air set point = 60°F, ASHRAE Class A2

Table 6.2 gives the typical response times of instruments used in this control methodology over the three trials of control. The MATLAB code that received all of the relevant data looped every 30 seconds in order to obtain the most recent data while still maintaining a connection to the BMS. However, a majority of the average response times well exceeded this 30-second loop. Typical values used in the calculation of the new outside air damper percentage, which included the return air and outside air temperatures, featured faster responses in the range of two minutes, with minimum response times of ~20 seconds which fits within the 30 second loop window.

`

Table 6.2 – Typical response times of control variables for Case Study 2

Variable	Average Response Time (hh:mm:ss)	Minimum Response Time (hh:mm:ss)
Outside Air Damper %	00:00:45	00:00:01
Supply Air Temperature	00:00:40	00:00:01
Mixed Air Temperature	00:18:00	00:02:36
Return Air Temperature	00:02:40	00:00:01
Outside Air Temperature	00:02:15	00:00:22
Space Temperature	00:30:00	00:02:24
Mixed Air Humidity	00:11:00	00:02:21
Outside Air Humidity	00:01:00	00:00:25
Space Humidity	00:20:00	00:02:16
VSD Speed	08:00:00	07:51:55

Conditions corresponding to the ASHRAE limits featured much longer response times, in the range of several minutes with minimum response times in the range of approximately two minutes. Of note, while these corresponded to longer request times the data itself did not vary significantly for these times. This phenomenon is best illustrated by the VSD speed, which was set as a constant 30% and the average response time was approximately eight hours. A possible explanation for this phenomenon is the PI

`

System's use of exception and compression [27]. Exception filters noise based on the deviation from a recorded value and the maximum time where a new value must recorded. Compression filters the trend of the data based on the deviation of the trend and the maximum time where a new value is recorded. Looking into the settings, these points were configured to have a maximum exception time of 10 minutes and a maximum compression time of 8 hours. This provides some explanation as to why variables, particularly those with low data variability, have much longer response times.

These trials demonstrate that overall use of economizers results in an estimated reduction of energy consumed by the chiller in the range of 1-2 kW. The fan power consumption typically dominated when the economizer was in use, while chiller power consumption dominated when the economizer was not in use or utilized less.

6.4 Summary

This case study provides a solid basis for expanding into economizer control to maintain a space within the specified ASHRAE class standards, which may vary depending on the application by the user. The mode of operation of this controller uses the economizer whenever applicable due to the control being based off of a mixed air target temperature near that of the supply air temperature set point. This controller results in a lower variation of the space temperature compared to typical economizer operation, and prevents wasted energy from cooling outside air that is warmer than the return air. Trials of utilizing this controller illustrate reduced energy consumption by the chiller whenever the economizer was utilized.

`

CHAPTER 7. CONCLUDING REMARKS

The overarching goal of this thesis was to approach the demands for the modern data center through thermal management and energy-saving techniques while maintaining space conditions that are conducive to the operation of servers and other IT equipment. This thesis was designed to devise a method to enact controls with cooling infrastructure to reduce user downtime in calculating optimal settings and manually changing the settings accordingly. The desired solution was to integrate the existing infrastructure of the PI System and the BMS for data collection and controls with the trending and conditional capabilities of MATLAB. This was done by importing and exporting the data between the PI System and MATLAB using PI AF SDK. With MODBUS communication, overwriting certain data points resulted in changes of the set points of the BMS. Additionally, PI Vision displays allowed users to more effectively monitor the conditions of the data centers and the controllers.

Case Study 1 focuses on the thermal management by proposing a controller that uses dynamic controls with heuristic temperature predictions to calculate an optimal CRAC temperature set point. The result is a controller that reaches a target server inlet air temperature within 30 minutes.

Case Study 2 uses the energy-saving technology of airside economization in the context of ASHRAE A1-A4 standards for data centers, with different temperature and humidity limits. While the A3 and A4 standards were designed for economizer use, the case study proposes a controller that uses the economizer whenever applicable to reach a target mixed air temperature near the supply air temperature set point. The result of using

`

the controller is in little variation of the space temperature, typically less than 2°C during normal operation. Additionally, the controller utilizes the economizer when appropriate, resulting in an estimated reduction of energy consumed by ~1-2 kW. Additionally, conditional statements ensure that the space is maintained within the limits prescribed by ASHRAE.

7.1 Key Contributions

The primary contribution of this thesis is the presented control methodology that bridges the gap between using controls and heuristic temperature prediction, while previous research focused on one aspect or the other. By coupling these research areas together, this thesis creates a prediction and control methodology that can be coupled with different instruments and scales of the data center environment, which in turn results in dynamic, smart cooling of the data center environment.

This thesis utilizes GUI displays through the creation of the PI Vision displays to monitor the CRAC infrastructure and the rack inlet temperatures. While BMS infrastructures may or may not include such GUI displays, having these displays for the server racks and cooling infrastructure allow for more intuitive monitoring of the conditions of the data center. As the heat densities of data centers and the computational capabilities of the servers is constantly increasing, it is important for engineers to be able to access and easily understand the full operating conditions of the data center. It will be increasingly important to utilize some sort of GUI display in data center monitoring, and future studies in the thermal and energy management of data centers should utilize such displays.

`

Finally, the approach of control and monitoring offered by this thesis and monitoring can easily fit in with other data center environments. Different data centers will of course have different infrastructure, such as custom BMS systems, different methods of control, and different methods of collecting and storing data, and as such the this methodology was designed to be somewhat open-ended to reflect the capability of modification for different data center environments. For instance, controls could be integrated with a GUI through National Instruments LabView system to effectively monitor and control data center infrastructure. Furthermore, instead of the heuristic temperature prediction method users could instead opt for a machine learning algorithm.

In all, this thesis illustrates new approaches with dynamic, user-defined controls and monitoring. Using the control methods in different case studies resulted in a higher degree of temperature control and energy efficiency, and using such techniques will become more critical in more effectively reducing the energy demand of data centers.

7.2 Recommendations for Future Work

This framework is based around data collection from the PI System and reading and writing of data from MATLAB based on heuristic temperature predictions. While this proved to be quite effective for this thesis, in the future I would likely utilize a temperature prediction model based around proper orthogonal decomposition or machine learning models, given the wide availability and accessibility of data within the DCL. Doing so would result in a reduction in the temperature prediction error and would rely less on the system identification experiments such as the TCI calibration.

`

The control methodologies utilized in this thesis only controlled temperature set point and economizer damper position at the room level, however this approach may be applied to other controllable instruments in a data center environment. For instance, the TCI relation in the first case study was used on a room level to reach a target temperature of one specific rack. By taking a composite TCI, this control methodology could be used to reach an average rack inlet temperature to result in more control at the overall room level. Similarly, using individual rack TCI values with more localized cooling instruments, this method could be scaled to the rack level. Coupled with control of different instruments such as the CRAC supply fan would result in a higher degree of control of the data center. Nonetheless, the principles of this more dynamic control are illustrated in this thesis.

7.2.1 Fault Detection, Catastrophic Events, and Data Security

As mentioned in the thesis preview in Chapter 1, the control methodology proposed in this thesis did not take into account catastrophic events. In terms of general monitoring, the conditional-based monitoring devised in the first case study presented simple status messages that corresponded to a user-defined temperature limits for the rack. This approach could similarly be applied to any temperature or humidity limits of the cooling infrastructure. Similarly, the conditionals used in the second case study will close the economizer dampers when the outside air conditions will cause the ASHRAE limits of the data center space to be exceeded. Such conditionals could be applied to other aspects of CRAC control to ensure that space conditions are maintained.

`

For more critical events such as power outages, the BMS system of the data center is the first line of defense. The BMS of a data center is designed to react to losses of cooling or power by bringing other equipment online and provide detailed notifications for data center operators to address the issue [4]. For instance, the BMS at the DCL sends email alerts to selected individuals when alarms are triggered.

The PI System has tools such as PI AF analyses and event frame generation that can effectively monitor the equipment of the data center. For instance, eBay monitors all of their equipment using the PI System, and use the historical data stored in the PI System with a root cause analysis to determine the source of outages (S. Robertson, personal communication, May 2, 2019). Adding further redundant alarms monitoring the data from the BMS using the tools in the PI System would also act as a further safeguard in the event that the BMS itself goes offline, giving operators detailed notifications to troubleshoot the issue.

Finally, one must consider the impact of data security when using any control methodology in a data center environment. BMS systems are designed to make use of robust network security, user password protocol, and controlled inbound and outbound network traffic to ensure the safety of the control of the cooling infrastructure of the data center [4]. For instance, some security measures of the BMS at the DCL include password protection to get to any controllable feature, as well as specific limits to the control set points.

For the PI System, attributes by default are configured as read-only. However, given the control methodology presented in this thesis, users must define practices to

`

ensure the security of the data center while using such controls. There are a few safety measures in place using the control methodology: the remote desktop on which the PI System is hosted is password protected, administrator approval is needed in order to make changes to the machine, and the MATLAB code will terminate if the connection to the computer is dropped. It is up to the data center operators to make use of security measures in the collection of data, especially in the control methodologies of the data center.

7.2.2 Case Study 1

To address the issues it would be beneficial to utilize other aspects of PID control, namely integral and derivative control. Derivative and integral control could potentially be implemented by taking both the derivative and integral of the linear fit of the historical data, which could result in improved settling time and overshoot.

The methodology of current and forecasted temperature difference as a basis of control could be paired with more sophisticated temperature prediction models to reduce overshoot, steady-state error, and settling time. Furthermore, this methodology may be applied to other aspects of thermal management of data centers, including controlling CRAC fan speeds, vent tile openings, and economization

7.2.3 Case Study 2

Controlling the temperature set point of the CRAC unit would result in more dynamic use of the economizer, as the economizer would adjust to a changing temperature set point. Furthermore, the conditional statements for the temperature and

`

humidity limits provide a framework in which the temperature set point control is adjusted to maintain the space conditions while reducing the power consumption. While this would result in higher variation of the space conditions, such a framework could lead to a substantial decrease in the chiller power. This case study provides a solid basis for expanding into such control to maintain a space within the ASHRAE class standards, as opposed to utilizing the economizer according to a certain supply air set point and then having the user define the set point to maintain the space condition.

Due to the limitation of not having humidification or dehumidification with the CRAC utilized in the second case study, the only method of control for the relative humidity of the space is by controlling the outside air damper percentage of the economizer. Using humidification and dehumidification along with the economizer could result in higher utilization of the economizer as these instruments would condition the outside air to be within the ASHRAE data center room conditions. However, the potential power consumption savings of higher economizer utilization must be weighed against additional power consumption of the humidification and dehumidification instruments.

`

APPENDIX A. MATLAB IMPORT FUNCTION

```
function [values_vec,times_vec] = AFSDK_import(element, attribute,
data_type, start_time, end_time)
%This function takes strings for a specific element and attribute for a
%given start time and end time (expressed as strings using AF time
%notation) and gives vectors of values, timestamps associated with the
time
%range


%The following lines load the assembly using PI AF SDK
afsdk = NET.addAssembly('OSIsoft.AFSDK');

import OSIsoft.AF.*

import OSIsoft.AF.Asset.*

import OSIsoft.AF.Time.*

import System.*

af_srvs = PISystems;

af_svr = af_srvs.Item('PIAF-GTECH'); %PI Server

af_db = af_svr.Databases.Item('Georgia Tech Data Center'); %Database

% af_db = af_svr.Databases.Item('AnalyticsTest'); %Database

switch lower(data_type) %Creates a list of attributes corresponding to
element and attribute name based on data type
    case 'single'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Single, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'double'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Double, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'boolean'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Boolean, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int16'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
```

```
`


            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int16, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int32'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int32, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int64'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int64, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
end

time_range = AFTimeRange(start_time,end_time); %Defines the time range

my_attribute = af_attr_list.Item(0); %Obtains the attribute

af_values = my_attribute.GetValues(time_range, 0,
my_attribute.DefaultUOM); %Creates an assembly of values for the
attribute for the given time range

for i = 0:(af_values.Count-1) %MATLAB vectors index from 1, while
assembly indexes from 0

    values_vec(i+1) = af_values.Item(i).Value; %Creates a vector of
values for the attribute from the assembly

    a = af_values.Item(i).Timestamp.UtcSeconds; %Obtains the timestamp
from the assembly, in UTC seconds format

    t = datenum(datetime(a, 'ConvertFrom','posixtime')); %Converts UTC
seconds format into MATLAB serial date number

    times_vec(i+1) = t; %Creates a vector of timestamps corresponding
to the attribute from the assembly

    end

end
```

`

APPENDIX B. MATLAB WRITER FUNCTION

```
function AFSDK_writer(element,attribute,data_type,output)
%This function takes strings for a specific element and attribute to
write
%strings generated from MATLAB to PI AF

%The following lines load the assembly using PI AF SDK
afsdk = NET.addAssembly('OSIsoft.AFSDK');

import OSIsoft.AF.*

import OSIsoft.AF.Asset.*

import OSIsoft.AF.Time.*

import System.*

af_srvs = PISystems;

af_svr = af_srvs.Item('PIAF-GTECH'); %PI Server

af_db = af_svr.Databases.Item('Georgia Tech Data Center'); %Database

% af_db = af_svr.Databases.Item('AnalyticsTest'); %Database

switch lower(data_type) %Creates a list of attributes corresponding to
element and attribute name based on data type
    case 'single'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Single, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'double'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Double, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'boolean'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Boolean, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int16'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int16, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int32'
```

`

```
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int32, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int64'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int64, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'string'
        output = string(output);
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.String, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
end

my_attribute = af_attr_list.Item(0); %Obtains the attribute

new_afval = AFValue(my_attribute, output, AFTime.Now); %Defines the
attribute's value at the current time

my_attribute.SetValue(new_afval); %Writes the attribute data to PI AF

end
```

`

APPENDIX C. MATLAB CONTROLLER FOR CASE STUDY 1

```
startTime = '*-5m'; %Start time for forecast period
endTime = '*'; %End time for forecast period
codeRunning = true; %Initiates loop
target_temp = 60; %Target temperature for WTS units

TCI = 0.95; %Thermal correlation index

CRAC = 'CRAC 5'; %CRAC unit corresponding to changes

while codeRunning == true %Infinite loop to constantly update values
%% WM8
   [WM8_J1_values_vec,WM8_J1_times_vec] =
AFSDK_import('WM8','J1','double',startTime, endTime);
        %Finds a specific attribute and outputs vectors for timestamps
and values for the given time range
    [WM8_J1_future_values,WM8_J1_future_times]
=AFSDK_forecaster(WM8_J1_values_vec,WM8_J1_times_vec);
        %Gives future values/timestamps
    [WM8_J1_tempStatus, WM8_J1_forecastStatus] =
AFSDK_forecast_conditionals(WM8_J1_future_values,WM8_J1_future_times);
        %Gives status strings on temperature and forecast status
    AFSDK_forecast_writer('WM8','J1
Forecast','double',WM8_J1_future_times, WM8_J1_future_values) %Writes
forecast data back to PI AF
    AFSDK_writer('WM8','J1 Temperature
Status','string',WM8_J1_tempStatus) %Writes temperature status back to
PI AF
    AFSDK_writer('WM8','J1 Forecast
Status','string',WM8_J1_forecastStatus) %Writes forecast status back to
PI AF
    WM8_J1_CRAC_temp_out = AFSDK_TCI(WM8_J1_values_vec,
WM8_J1_future_values, target_temp, CRAC, TCI); %Calculates the optimal
CRAC Temperature Set Point

   [WM8_J2_values_vec,WM8_J2_times_vec] =
AFSDK_import('WM8','J2','double',startTime, endTime);
        %Finds a specific attribute and outputs vectors for timestamps
and values for the given time range
    [WM8_J2_future_values,WM8_J2_future_times]
=AFSDK_forecaster(WM8_J2_values_vec,WM8_J2_times_vec);
        %Gives future values/timestamps
    [WM8_J2_tempStatus, WM8_J2_forecastStatus] =
AFSDK_forecast_conditionals(WM8_J2_future_values,WM8_J2_future_times);
        %Gives status strings on temperature and forecast status
    AFSDK_forecast_writer('WM8','J2
Forecast','double',WM8_J2_future_times, WM8_J2_future_values) %Writes
forecast data back to PI AF
    AFSDK_writer('WM8','J2 Temperature
Status','string',WM8_J2_tempStatus) %Writes temperature status back to
PI AF
    AFSDK_writer('WM8','J2 Forecast
Status','string',WM8_J2_forecastStatus) %Writes forecast status back to
```

```
`


PI AF
    WM8_J2_CRAC_temp_out = AFSDK_TCI(WM8_J2_values_vec,
WM8_J2_future_values, target_temp, CRAC, TCI); %Calculates the optimal
CRAC Temperature Set Point

    [WM8_J3_values_vec,WM8_J3_times_vec] =
AFSDK_import('WM8','J3','double',startTime, endTime);
        %Finds a specific attribute and outputs vectors for timestamps
and values for the given time range
    [WM8_J3_future_values,WM8_J3_future_times]
=AFSDK_forecaster(WM8_J3_values_vec,WM8_J3_times_vec);
        %Gives future values/timestamps
    [WM8_J3_tempStatus, WM8_J3_forecastStatus] =
AFSDK_forecast_conditionals(WM8_J3_future_values,WM8_J3_future_times);
        %Gives status strings on temperature and forecast status
    AFSDK_forecast_writer('WM8','J3
Forecast','double',WM8_J3_future_times, WM8_J3_future_values) %Writes
forecast data back to PI AF
    AFSDK_writer('WM8','J3 Temperature
Status','string',WM8_J3_tempStatus) %Writes temperature status back to
PI AF
    AFSDK_writer('WM8','J3 Forecast
Status','string',WM8_J3_forecastStatus) %Writes forecast status back to
PI AF
    WM8_J3_CRAC_temp_out = AFSDK_TCI(WM8_J3_values_vec,
WM8_J3_future_values, target_temp, CRAC, TCI); %Calculates the optimal
CRAC Temperature Set Point

    [WM8_J4_values_vec,WM8_J4_times_vec] =
AFSDK_import('WM8','J4','double',startTime, endTime);
        %Finds a specific attribute and outputs vectors for timestamps
and values for the given time range
    [WM8_J4_future_values,WM8_J4_future_times]
=AFSDK_forecaster(WM8_J4_values_vec,WM8_J4_times_vec);
        %Gives future values/timestamps
    [WM8_J4_tempStatus, WM8_J4_forecastStatus] =
AFSDK_forecast_conditionals(WM8_J4_future_values,WM8_J4_future_times);
        %Gives status strings on temperature and forecast status
    AFSDK_forecast_writer('WM8','J4
Forecast','double',WM8_J4_future_times, WM8_J4_future_values) %Writes
forecast data back to PI AF
    AFSDK_writer('WM8','J4 Temperature
Status','string',WM8_J4_tempStatus) %Writes temperature status back to
PI AF
    AFSDK_writer('WM8','J4 Forecast
Status','string',WM8_J4_forecastStatus) %Writes forecast status back to
PI AF
    WM8_J4_CRAC_temp_out = AFSDK_TCI(WM8_J4_values_vec,
WM8_J4_future_values, target_temp, CRAC, TCI); %Calculates the optimal
CRAC Temperature Set Point

    CRAC_temp_out = mean([WM8_J1_CRAC_temp_out, WM8_J2_CRAC_temp_out,
WM8_J3_CRAC_temp_out, WM8_J4_CRAC_temp_out]); %Calculates composite
value for CRAC Temperature Set Point
    CRAC_temp_out = round(CRAC_temp_out);
    AFSDK_writer('CRAC 5','Temperature Set Point
Write','double',CRAC_temp_out) %Writes CRAC Temperature Set Point back
to PI AF
```

```
    AFSDK_writer('WM8','Code Status','boolean',codeRunning) %Writes
code status back to PI AF
%%
    pause(30) %Used to control the frequency of calculations and
forecasts
end
```

`

APPENDIX D. MATLAB FORECASTER FUNCTION

```
function [future_values,future_times] =
AFSDK_forecaster(values_vec,times_vec)
%This function takes the imported values and timestamps from the
database
%and creates a vector of future times and values based off a first
order
%polynomial fit

dayDt = 1; %Change in time for a day using serial number notation
monthDt = dayDt.*31; %Change in time for a month using serial number
notation
yearDt = dayDt.*365; %Change in time for a year using serial number
notation
hourDt = dayDt./24; %Change in time for an hour using serial number
notation
minuteDt = hourDt./60; %Change in time for a minute using serial number
notation
secondDt = minuteDt./60; %Change in time for a second using serial
number notation

[polynomial,S,MU] = polyfit(times_vec,values_vec,1); %Performs a first
order polynomial fit on the specified range

[endtime_year, endtime_month, endtime_day, endtime_hour,
endtime_minute, endtime_second] = ...
    datevec(times_vec(end)); %Obtains a date vector of the last
timestamp of the imported data

if endtime_second >= 30
    endtime_minute = endtime_minute+1; %Rounds the time scale to the
nearest minute for the forecast time scale
end

endtime_second = 0; %Sets the forecast time scale to the minute basis

end_time = datenum([endtime_year, endtime_month, endtime_day,
endtime_hour, endtime_minute, endtime_second]);
%Re-converts the last timestamp of the imported data back to serial
date number, now rounded to the nearest minute

future_times = end_time:minuteDt:(end_time+5.*minuteDt); %Creates a
vector of future times 30 minutes into the future

[future_values, DELTA] = polyval(polynomial,future_times,S,MU);
%Creates a vector of values based on the future times and the
polynomial fit

end
```

`

APPENDIX E. MATLAB FORECAST WRITER FUNCTION

```
function
AFSDK_forecast_writer(element,attribute,data_type,future_times,
future_values)
%This function takes strings for a specific element and attribute to
write
%data to PI AF as future time, along with the future times and future
values that
%correspond to the forecast

%The following lines load the assembly using PI AF SDK
afsdk = NET.addAssembly('OSIsoft.AFSDK');

import OSIsoft.AF.*

import OSIsoft.AF.Asset.*

import OSIsoft.AF.Time.*

import System.*

af_srvs = PISystems;

af_svr = af_srvs.Item('PIAF-GTECH'); %PI Server

af_db = af_svr.Databases.Item('Georgia Tech Data Center'); %Database

% af_db = af_svr.Databases.Item('AnalyticsTest'); %Database

switch lower(data_type) %Creates a list of attributes corresponding to
element and attribute name based on data type
    case 'single'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Single, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'double'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Double, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'boolean'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Boolean, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int16'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
```

```
`


            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int16, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int32'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int32, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
    case 'int64'
        af_attr_list = AFAttribute.FindElementAttributes(af_db, [],
element, [], ...
            [], AFElementType.Any, attribute, [], ...
            TypeCode.Int64, true, AFSortField.Name,
AFSortOrder.Ascending, 1000);
end

my_attribute = af_attr_list.Item(0); %Obtains the attribute

for i = 1:length(future_values)

    t = datetime(datevec(future_times(i))); %Converts MATLAB serial
time into datetime format

    a = posixtime(t); %Converts datetime format into UTC seconds

    new_afval = AFValue(my_attribute, future_values(i), AFTime(a));
%Defines the attribute's value at a specific time

    my_attribute.SetValue(new_afval); %Writes the attribute data to PI
AF

end

end
```

`

APPENDIX F. MATLAB FORECAST CONDITIONAL FUNCTION

```
function [tempStatus, forecastStatus] =
AFSDK_forecast_conditionals(future_values,future_times)
%This function takes the future values and time stamps, along with the
%temperature limit, to create status strings for the temperature status
and
%forecast status
tempLimit = 95;

forecastTest = future_times(future_values>=tempLimit); %Creates a test
to see if the forecast goes above the accepted temperature limit

if future_values(1) >= tempLimit %Finds if current temperature is over
close to the temperature limit

        tempStatus = sprintf('CRITICAL: Take immediate action.\nCurrent
temperature of %.2f deg F exceeds %.2f deg F temperature limit.\n',...
            future_values(1), tempLimit); %Error message for exceeded
temperature limit

        forecastStatus = sprintf('Forecast status unavailable,
temperature limit is exceeded.'); %Forecast message if temperature
limit is exceeded

elseif future_values(1) >= 0.95.*tempLimit %Finds if current
temperature is too close to the temperature limit

        tempStatus = sprintf('WARNING: Take corrective action.\nCurrent
temperature of %.2f deg F is within 5%% of %.2f deg F temperature
limit.\n',future_values(1), tempLimit); %Error message for high
temperature

        if length(forecastTest) > 0 %Used to predict the time at which
the temperature limit will be reached, using a more aggressive
            %forecasting due to the higher temperature

            violationTime = forecastTest(1); %Identifies the time at
which the forecast violates the temperature limit

            [violationTime_year, violationTime_month,
violationTime_day, violationTime_hour, violationTime_minute,
violationTime_second] = ...
                datevec(violationTime); %Date vector corresponding to
the time where the temperature limit is exceeded

            violationTime_second = round(violationTime_second); %Rounds
the violation time to the nearest second

            [~,~,~,timediff_hour, timediff_minute, timediff_second] =
datevec((violationTime-times_vec(end)));
                %Time difference between the current time and the
violation time
```

```
`


                forecastStatus = sprintf('WARNING: Take corrective
action.\nTemperature is forecasted to exceed temperature limit of %.02f
deg F in %d hours and %d minutes at %02d:%02d:%02d GMT on
%02d/%02d/%02d.\n',tempLimit,timediff_hour,timediff_minute,violationTim
e_hour,...
                    violationTime_minute, violationTime_second,
violationTime_month, violationTime_day,...
                    violationTime_year) %Error message for forecast
violation

        else
                forecastStatus = sprintf('Forecast Status Normal.');
%Forecast message in case none of the following conditional statements
apply

        end

else

    tempStatus = sprintf('Temperature Status Normal.'); %Message for
normal temperature

    if length(forecastTest) > 0 %Used to predict the time at which the
temperature limit will be reached, using a more aggressive
            %forecasting due to the higher temperature

            violationTime = forecastTest(1); %Identifies the time at
which the forecast violates the temperature limit

            [violationTime_year, violationTime_month,
violationTime_day, violationTime_hour, violationTime_minute,
violationTime_second] = ...
                datevec(violationTime); %Date vector corresponding to
the time where the temperature limit is exceeded

            violationTime_second = round(violationTime_second); %Rounds
the violation time to the nearest second

            [~,~,~,timediff_hour, timediff_minute, timediff_second] =
datevec((violationTime-times_vec(end)));
                %Time difference between the current time and the
violation time

                forecastStatus = sprintf('WARNING: Take corrective
action.\nTemperature is forecasted to exceed temperature limit of %.02f
deg F in %d hours and %d minutes at %02d:%02d:%02d GMT on
%02d/%02d/%02d.\n',tempLimit,timediff_hour,timediff_minute,violationTim
e_hour,...
                    violationTime_minute, violationTime_second,
violationTime_month, violationTime_day,...
                    violationTime_year) %Error message for forecast
violation

        else
                forecastStatus = sprintf('Forecast Status Normal.');
```

```
`


%Forecast message in case none of the following conditional statements
apply

        end

end
end
```

`

APPENDIX G. MATLAB TCI FUNCTION

```
function [CRAC_temp_out] = AFSDK_TCI(values_vec,future_values,
target_temp, CRAC, TCI)
% %This function takes the forecast of future values and uses a target
% temperature, the read value of the CRAC Temperature Set Point, and
the
% Thermal Correlation Index (TCI) to determine the new CRAC Temperature
Set
% Point

current_temp = values_vec(end); %Current temperature

Return_Air_Actual = AFSDK_import(CRAC,'Return Air','double', '*', '*');
%Imports return air temperature

Return_Air_Setpoint = AFSDK_import(CRAC,'Temperature Set Point
Read','double', '*', '*'); %Imports return air temperature setpoint

if CRAC == 'CRAC 5' %Accounts for data cutoff that results in the
setpoint being 1 F lower than the value read in the BMS
    Return_Air_Setpoint = Return_Air_Setpoint + 1;
elseif CRAC == 'CRAC 6'
    Return_Air_Setpoint = Return_Air_Setpoint + 1;
end

delta_T1 = target_temp - current_temp; %Current temperature difference

delta_T2 = target_temp - future_values(end); %Forecast temperature
difference

if abs(delta_T1)<=5 %Sets control based on current temperature
difference if
% temperature is within 5 F of the target, otherwise uses forecast
control

    delta_T=delta_T1;
else

    delta_T=delta_T2;
end

CRAC_temp_in = Return_Air_Setpoint; %Imported CRAC return air
temperature

CRAC_temp_out = (delta_T./TCI) + CRAC_temp_in;  %Exported CRAC return
air temperature

if abs(delta_T1) <= 1 %Exported CRAC return air temperature if current
temperature
    %difference is within 1 F
```

```
`

    CRAC_temp_out = CRAC_temp_in;

end

if CRAC_temp_out > 85 %Sets upper writing limit

    CRAC_temp_out = 85;

elseif CRAC_temp_out < 40 %Sets lower writing limit

    CRAC_temp_out = 40;

end

end
```

`

APPENDIX H. MATLAB CONTROLLER FOR CASE STUDY 2

```
class = 'a2';

switch lower(class) %Defines the temperature and humidity limits based
on
%ASHRAE class

    case{'a'}
        lowTemp = 64.4;

        highTemp = 80.6;

        lowHum = 44.5;

        highHum = 60;

    case{'a1'}
        lowTemp = 59;

        highTemp = 89.6;

        lowHum = 20;

        highHum = 80;

    case{'a2'}
        lowTemp = 50;

        highTemp = 89.6;

        lowHum = 20;

        highHum = 80;

    case{'a3'}
        lowTemp = 41;

        highTemp = 104;

        lowHum = 8;

        highHum = 85;

    case{'a4'}
        lowTemp = 41;

        highTemp = 113;

        lowHum = 8;
```

```
`

        highHum = 90;

    case{'b'}
        lowTemp = 41;

        highTemp = 89.6;

        lowHum = 8;

        highHum = 80;

    case{'c'}
        lowTemp = 41;

        highTemp = 104;

        lowHum = 8;

        highHum = 80;

end

codeRunning = true; %Initiates loop

CRAC = 'CRAC 2'; %CRAC unit corresponding to changes

while codeRunning == true %Infinite loop to constantly update values

    [MA_temp,~] = AFSDK_import(CRAC,'Mixed Air Temperature','double',
'*', '*');
    %Current mixed air temperature

    [MA_hum,~] = AFSDK_import(CRAC,'Mixed Air Humidity','double', '*',
'*');
    %Current mixed air humidity

    [SA_temp,~] = AFSDK_import(CRAC,'Supply Air','double', '*', '*');
    %Current supply air temperature

    [SA_hum,~] = AFSDK_import(CRAC,'Supply Air Humidity','double', '*',
'*');
    %Current supply air humidity

    [RA_temp,~] = AFSDK_import(CRAC,'Return Air','double', '*', '*');
    %Current return air temperature

    [OA_temp,~] = AFSDK_import(CRAC,'Outside Air Temperature','double',
'*', '*');
    %Current outside air temperature

    [OA_hum,~] = AFSDK_import(CRAC,'Outside Air Humidity','double',
'*', '*');
    %Current outside air humidity
```

```
    [space_temp,~] = AFSDK_import(CRAC,'Space Temperature','double',
'*', '*');
    %Current space air temperature

    [space_hum,~] = AFSDK_import(CRAC,'Space Humidity','double', '*',
'*');
    %Current space air humidity

    [OA_damper_in,~] = AFSDK_import(CRAC,'Economizer Damper Position
Read','double', '*', '*');
    %Current outside air economizer damper position

    OA_damper_in = OA_damper_in + 1;

    [SA_setpoint_in,~] = AFSDK_import(CRAC,'Temperature Set Point
Read','double', '*', '*');
    %Current supply air set point

    MA_target = SA_temp+2; %Sets target temperature of mixed air

    OA_damper_out = (100*(MA_target-RA_temp))/(OA_temp-RA_temp);
    %Calculates damper% based on mixed air target

    if space_temp > 0.95*highTemp  && OA_temp > MA_temp %Sets damper%
to 0
        %if space is within 5% of temperature limits and outside air
will
        %cause the limits to be exceeded

        OA_damper_out = 0;

    elseif space_temp < 1.05*lowTemp && OA_temp < MA_temp

        OA_damper_out = 0;

    end

    if space_hum > 0.95*highHum && OA_hum > MA_hum %Sets damper% to 0
        %if space is within 5% of humidity limits and outside air will
        %cause the limits to be exceeded

        OA_damper_out = 0;

    elseif space_temp < 1.05*lowTemp && OA_hum < MA_hum

       OA_damper_out = 0;

    end



    if OA_damper_out > 100 %Maximum damper% = 100%

        OA_damper_out = 100;
```

```
`


    elseif OA_damper_out < 0

        OA_damper_out = 0; %Maximum damper% = 0%

    end

AFSDK_writer(CRAC,'Economizer Damper Position
Write','double',OA_damper_out);
%Writes new damper position to PI AF

    pause(30) %Used to control the frequency of calculations and
forecasts
end
```

APPENDIX I. CASE STUDY 2 HUMIDITY VALUES

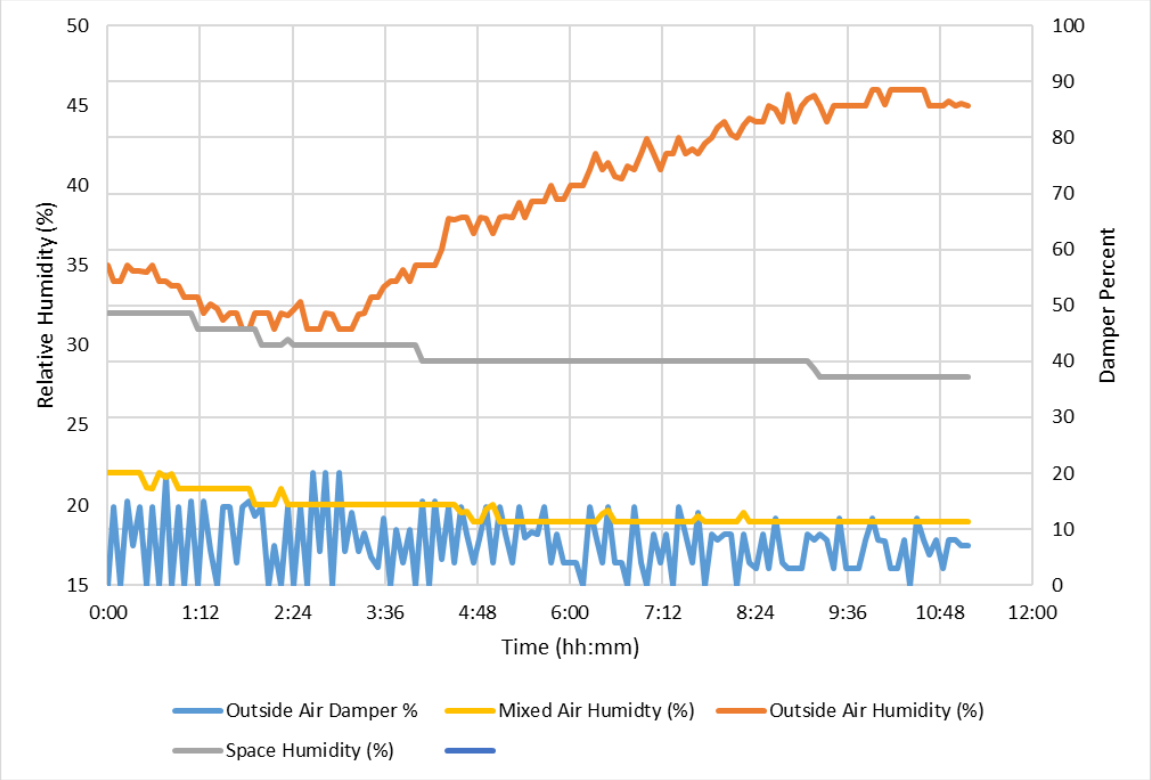

Figure A.1 – Economizer humidity with supply air set point = 65°F, ASHRAE Class A3

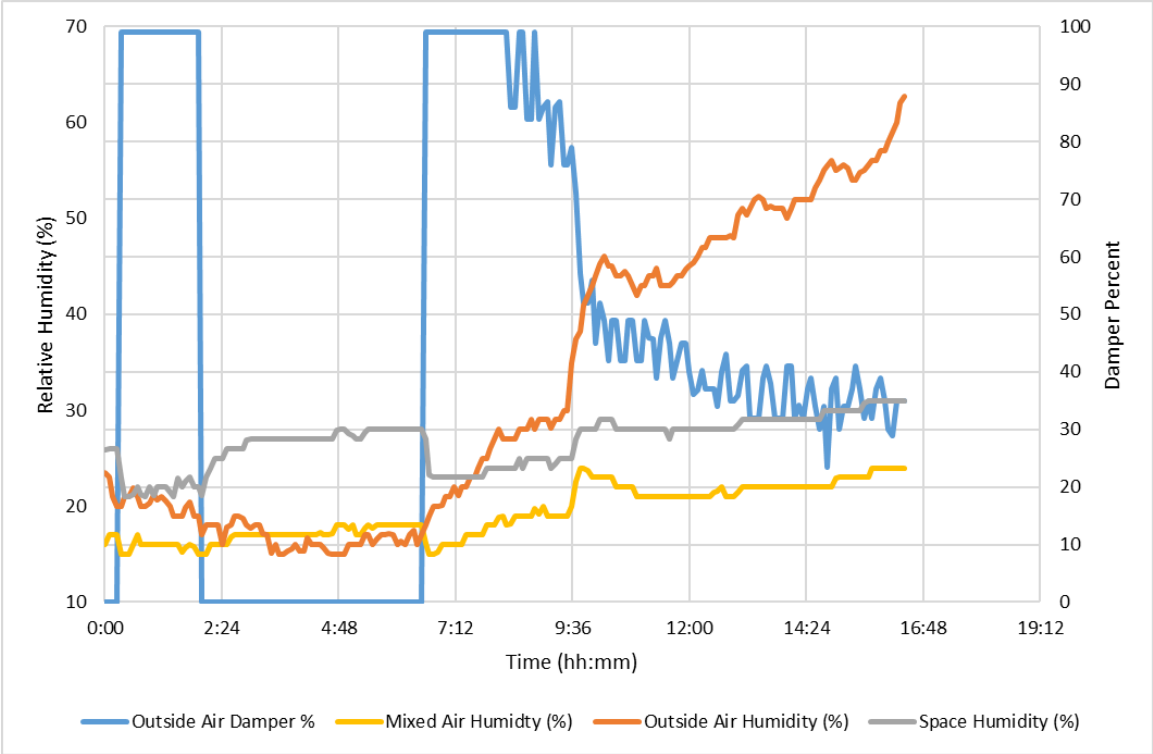

Figure A.2 – Economizer humidity with supply air set point = 55°F, ASHRAE Class A3

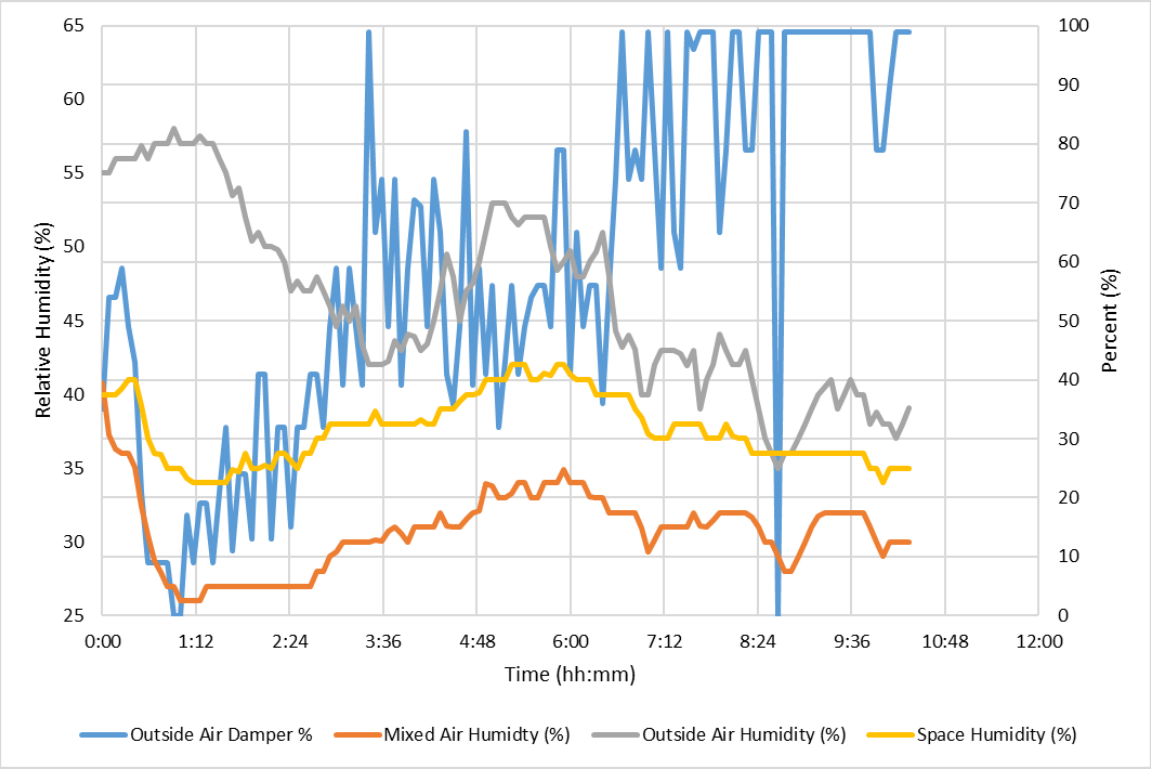

Figure A.3 – Economizer humidity with supply air set point = 60°F, ASHRAE Class A2

`

REFERENCES

[1] Y. Joshi, and P. Kumar, *Energy Efficient Thermal Management of Data Centers*. New York, NY: Springer, 2012.

[2] ASHRAE, “Data Center Power Equipment Thermal Guidelines and Best Practices (Whitepaper).” *Technical Committee 9.9 Mission Critical Facilities, Data Centers, Technology Spaces, and Electronic Equipment*, Atlanta, GA, 2016.

[3] ASHRAE “Thermal guidelines for data processing environments – expanded data center classes and usage guidance”, American Society for Heating, Refrigeration and Air Conditioning, Atlanta, Georgia, 2011.

[4] A. Smith, "Building Management Systems (BMS) Seminar 1 - The Basics Explained", Melbourne, AU.

[5] B. Anderson and P. Donovan, “Selecting a Building Management System (BMS) for Sites with a Data Center or IT Room (Whitepaper)”, *Schneider Electric*.

[6] L. Li, C.-J. M. Liang, J. Liu, S. Nath, A. Terzis, and C. Faloutsos, "Thermocast: A Cyber-Physical Forecasting Model for Datacenters," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, San Diego, California, USA, 2011.

[7] J. Chen, R. Tan, Y. Wang, G. Xing, X. Wang, X. Wang*, et al.*, "A High-Fidelity Temperature Distribution Forecasting System for Data Centers," in *IEEE 33rd Real-Time Systems Symposium*, pp. 215-224, 2012.

[8] E. Samadiani and Y. Joshi, "Proper Orthogonal Decomposition for Reduced Order Thermal Modeling of Air Cooled Data Centers," *ASME Journal of Heat Transfer,* vol. 132, pp. 071402-071402-14, 2010.

[9] E. Samadiani, Y. Joshi, H. Hamann, M. K. Iyengar, S. Kamalsy, and J. Lacey, "Reduced Order Thermal Modeling of Data Centers Via Distributed Sensor Data," *ASME Journal of Heat Transfer,* vol. 134, pp. 041401-041401-8, 2012.

[10] N. Ahuja, C. W. Rego, S. Ahuja, Z. Shen, and S. Shrivastava, "Real Time Monitoring and Availability of Server Airflow for Efficient Data Center Cooling," in

`

29th IEEE Semiconductor Thermal Measurement and Management Symposium, pp. 243-247, 2013.

[11] T. D. Boucher, D. M. Auslander, C. E. Bash, C. C. Federspiel, and C. D. Patel, "Viability of Dynamic Cooling Control in a Data Center Environment," *ASME Journal of Electronic Packaging,* vol. 128, pp. 137-144, 2005.

[12] C. Bash, C. D. Patel, and R. K. Sharma, "Dynamic Thermal Management of Air Cooled Data Centers," in *Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. ITHERM'06. The Tenth Intersociety Conference on*, pp. 8 pp.-452, 2006.

[13] S. Zhang, T. Zhou, N. Ahuja, G. Refai-Ahmed, Z. Yongzhong, C. Guofeng*, et al.*, "Real Time Thermal Management Controller for Data Center," in *Fourteenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm)*, pp. 1346-1353, 2014.

[14] C. D. Patel, C. E. Bash, R. Sharma, M. Beitelmal, and R. Friedrich, "Smart Cooling of Data Centers," in *ASME. International Electronic Packaging Technical Conference and Exhibition*, pp. 129-137, 2003.

[15] Z. Wang, C. Bash, C. Hoover, A. McReynolds, C. Felix, and R. Shih, "Integrated Management of Cooling Resources in Air-Cooled Data Centers," in *2010 IEEE International Conference on Automation Science and Engineering*, pp. 762-767, 2010.

[16] OSIsoft Learning, *OSIsoft: PI Basics- Map of the PI System,* Mar 7, 2014. Accessed on: June 23, 2018. [Video File]. Available: https://www.youtube.com/watch?v=zj2EoTkpz1k

[17] OSIsoft Learning, *OSIsoft: PI Basics- What are PI Assets, PI Attributes, and PI Tags?*, Mar 7, 2014. Accessed on: June 23, 2018. [Video File]. Available: https://www.youtube.com/watch?v=zj2EoTkpz1k

[18] S. Pilon, Y. Gauthier, Fattahi, A. and Ng, D. *White Paper - Using PI Data with MATLAB,* PI Developers Club, OSIsoft, San Leandro, CA, 2014. [online]. Available: https://pisquare.osisoft.com/servlet/JiveServlet/download/1305-4-13703/White%20Paper%20-%20Using%20PI%20Data%20with%20MATLAB.PDF.

[19] Y. Sverdlik, *Google is Switching to a Self-Driving Data Center Management System*, Data Center Knowledge, Aug. 2018. Accessed on: May 5, 2019. [Online].

`

Available: https://www.datacenterknowledge.com/google-alphabet/google-switching-self-driving-data-center-management-system

[20] *SN802GRC-4DM-X 4-Channel DIN-rail Mount Wi-Fi MODBUS Sensor Modem Technical Reference*. Murata Manufacturing Co, 2014.

[21] *Liebert Deluxe System/3™ - Chilled Water System Design Manual - 50 & 60Hz*. Emerson Electric Co, 2007.

[22] *CEETHERM Lab, Georgia Institute of Technology*. McKenney's Inc., 2009.

[23] *Duct Mounted Thermistor and RTD Temperature Probe*, BAPI, Gays Mills, WI, 2008.
[24] *Humidity & Combination Temp/Humidity Sensors*, BAPI, Gays Mills, WI, 2008.
[25] J. Athavale, (2019). *Artificial Neural Network Based Prediction and Cooling Energy Optimization of Data Centers*. Ph.D. Georgia Institute of Technology.

[26] "Series R Rotary Liquid Chiller 70 to 400 Tons Air-Cooled." Trane, La Crosse, WI, 1999.

[27] OSIsoft Learning, *OSIsoft: Exception and Compression Quick Summary,* Nov 13, 2012. Accessed on: May 12, 2019. [Video File]. Available: https://www.youtube.com/watch?v=6-scv3oQ7Kk